**Queen Mary, University of London**
**School of Electronic Engineering & Computer Science**

# REAL-TIME APPEARANCE-BASED GAZE TRACKING

Thesis submitted to University of London

in partial fullment of the

requirements for the degree of

Doctor of Philosophy

**Sertan Kaymak**

Supervisor : Dr. Ioannis Patras

January, 2015.

# Statement of originality

I, Sertan Kaymak, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third partys copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date:

Details of collaboration and publications: I did my work under the supervision of my supervisor Dr. Ioannis Patras.

# Abstract

Gaze tracking technology is widely used in Human Computer Interaction applications such as in interfaces for assisting people with disabilities and for driver attention monitoring. However, commercially available gaze trackers are expensive and their performance deteriorates if the user is not positioned in front of the camera and facing it. Also, head motion or being far from the device degrades their accuracy.

This thesis focuses on the development of real-time time appearance based gaze tracking algorithms using low cost devices, such as a webcam or Kinect. The proposed algorithms are developed by considering accuracy, robustness to head pose variation and the ability to generalise to different persons. In order to deal with head pose variation, we propose to estimate the head pose and then compensate for the appearance change and the bias to a gaze estimator that it introduces. Head pose is estimated by a novel method that utilizes tensor-based regressors at the leaf nodes of a random forest. For a baseline gaze estimator we use an SVM-based appearance-based regressor. For compensating the appearance variation introduced by the head pose, we use a geometric model, and for compensating for the bias we use a regression function that has been trained on a training set. Our methods are evaluated on publicly available datasets.

# Acknowledgments

I would like to thank my supervisor Dr. Ioannis Patras for his continuous guidance, suggestions, encouragement and feedback. I would also like to thank my co-supervisors Dr. Tao Xiang and Prof. Andrea Cavallaro for their feedback and guidances during my Phd study.

I would like to take this opportunity to thank Prof. Ebroul Izquierdo for his help during the period of time in the Multimedia and Vision Lab of Queen Mary University of London (MMV). I would also like to express my gratitude to my colleagues and the staff in MMV Lab.

Lastly, I would like to thank my family for their encouragement and support.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Human beings are exploring space by fixing their eyes on their focus of attention. This fixation causes subjective formation of visual space at the area of interest. Understanding the subjective formation of space can be achieved by developing interfaces to capture the subject's eye movements, [19]. The interfaces can also be used in Human Computer Interaction. For example, the method can be used in interfaces for computer input and these interfaces can be used instead of the traditional keyboards and mouses. Eye gaze direction might also reveal information about humans such as cognitive processes,


**Figure 1.1:** One of the gaze tracking system users who suffers from spinal cord injury, [1].

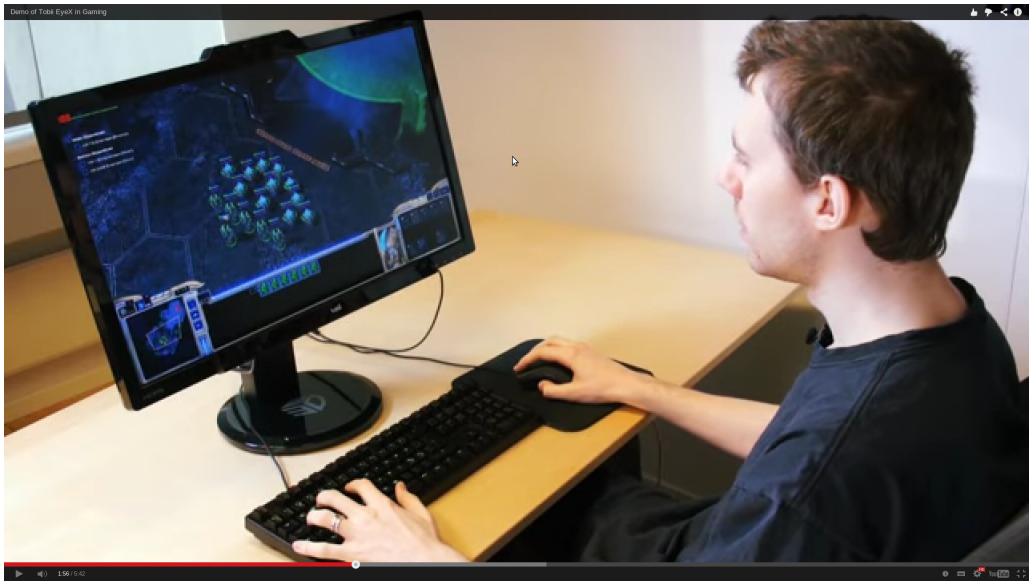



**Figure 1.2:** One of the users who is playing a game with assistive gaze tracking system, [1].

emotional states and their focus of attention [20]. However, currently available gaze tracking technologies have several constraints such as being person dependent, and work under limited head pose variation. In this thesis, an appearance-based gaze tracking method was developed to capture the subject's eye movements.

## 1.2 Applications

There are many applications of gaze tracking technology. First, the gaze tracking technology can be used as assistive technology for people who are physically limited in their activities. Specifically, it is very crucial for people who suffer from spinal cord injury [1]. For these people their eye movements are the only means of communication. Figure 1.1 shows one of the gaze tracking system for users who suffer from spinal cord injury. Second, gaze estimation technology can also be used for gaming [1] where the game environment changes according to the user's gaze direction. Figure 1.2 shows a person playing a game with an assistive gaze tracking system. Third, driver assistance systems [1, 21], might also be based on gaze tracking systems. Gaze tracking for the driver assistance systems are usually combined with head pose estimation [9, 22–25]. The head pose provides coarse gaze direction when the eye regions are occluded or not visible. The driver's focus of attention is detected using these systems in order to avoid possible collisions. It is well known that driver inattention and distraction are the main causes


**Figure 1.3:** One of the gaze driver assistance systems, [1].

of automotive collisions. Figure 1.3 shows one of the gaze driver assistance systems. Furthermore, the estimation of the gaze direction can also be used in graphic rendering applications [26]. For example, the renderer adjusts the quality of rendering region according to users region of attention. Moreover, gaze tracking technology can also be used in behavioural studies [26]. The frequency of looking at the display and the time period taken can be used for the analysis of efficiency of the display.

Gaze tracking technology might also be used for the development of algorithms which allow recognition of daily activities [2]. Activity recognition has many application areas such as human-computer interaction, and elderly care. One of the recent studies, [2], focused on daily activity recognition on egocentric videos. The study proposed to develop an algorithm which allows recognition of actions on egocentric videos using gaze estimates. In their study, a wearable gaze tracking system is used to relate actions and gaze locations on videos in order to learn activities. The learned models provide automatic estimation of an action region on videos. Figure 1.4 demonstrates the relationship between gaze location and the activity. Gaze location is defined by red circle on the video which is pouring milk into a cup. Finally, gaze tracking technology might be useful for estimating a strabismus angle of a person [5]. Strabismus is a condition where a person can not look at a fix point with both eyes. The occurrence of this condition is 2-5% in the populations. The misaligned focus of sight can be corrected by surgery, with a success rate of about 20-50%. The success rate of the surgery depends on: (i) accurate measurement of strabismus angle (20%), (ii) the surgical approach and success


**Figure 1.4:** Gaze location is defined by red circle on the action video which is pouring mike into cup [2]

of surgery (20%). Therefore, the estimation of the strabismus angle is a crucial factor for the successful correction of the focus of attention. A recent study, [5], proposed a method to obtain an estimate of strabismus angle accurately.

## 1.3 Challenges

However, automatic detection and estimation of eye movements is very challenging due to variation of human eyes, lighting conditions, occlusions and head pose variation. Figure 1.5 shows variation in the appearance of the eye regions. Advanced algorithms need to be developed for gaze estimation since the current methods are limited. One of the main limitations of current gaze estimation methods is that several cameras are required for estimating user‘s eye gaze [3, 4, 27–29]. Figure 1.6 shows a gaze tracking system consisting of several devices (three IR light sources and two video cameras). A several camera set-up is not easy to use since the calibration process has to be performed when the locations of cameras change. Methods which allow accurate gaze estimation from low cost single equipment (for example webcam, Kinect) need to be developed. Another limitation of current gaze estimation methods is that gaze estimation methods do not allow accurate gaze estimation while users move their heads naturally in front of the camera. Usually, the user has to keep their head still while using such gaze estimation systems. Although, previously proposed 3D gaze tracking methods allow

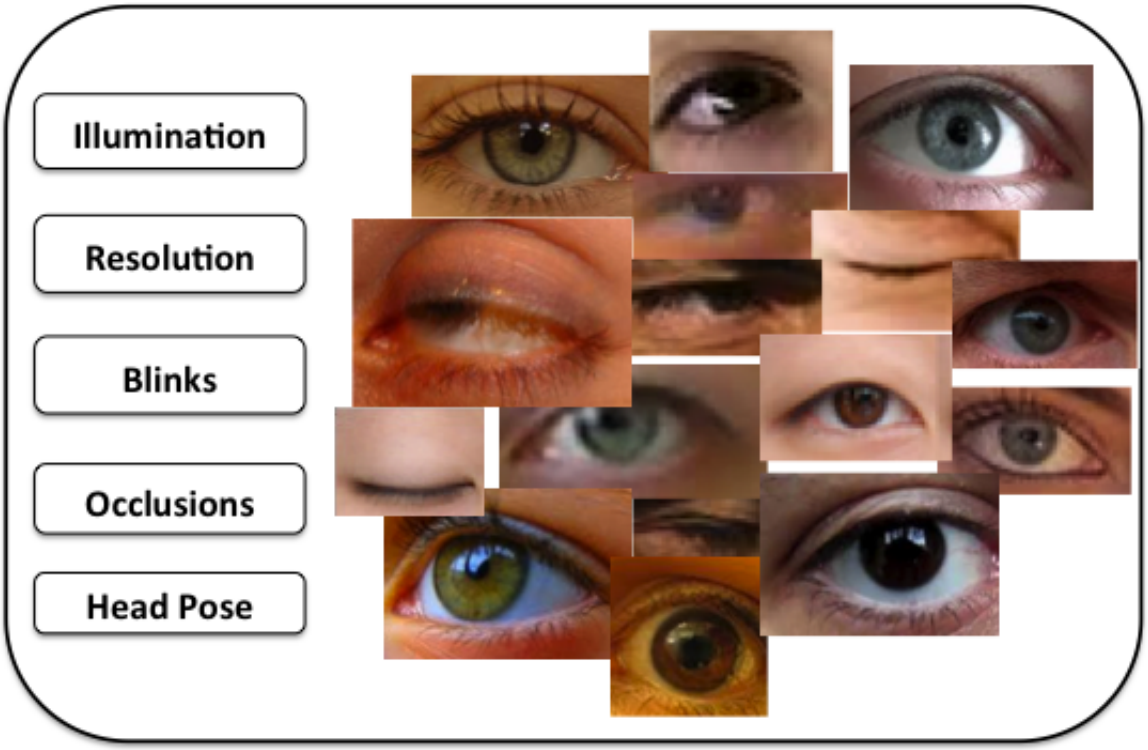



**Figure 1.5:** Variation in the appearance of eye regions.

more accurate gaze direction estimates under free head pose, the range of the user's head pose is limited. On the other hand, appearance based methods are not robust under the free head pose. The study in [29] has demonstrated that the gaze accuracy degrades when the head pose changes. The methods should be investigated more for eye gaze estimation under natural head pose. A brief overview of the approaches proposed for eye gaze estimation can be found in [20].

One of the limitations of appearance-based gaze tracking methods is that their performances degrade when a user moves with respect to a camera. The motion of a user might occur as a result of rotating his/her head. The rotation of the head causes changes in 2D eye images. Since regression methods are used to map eye images to the eye gaze points, the regression models can not deal with these changes efficiently. As a result, the user's head orientation can be calculated and the orientation parameters can be used to eliminate these changes on 2D eye images for more accurate gaze estimates. There are several head pose estimation methods: those that employ 2D images and others that employ 3D depth data. The methods that employ 3D data can be used for head pose parameter estimation using the recently developed RGB-D camera (Kinect). There are

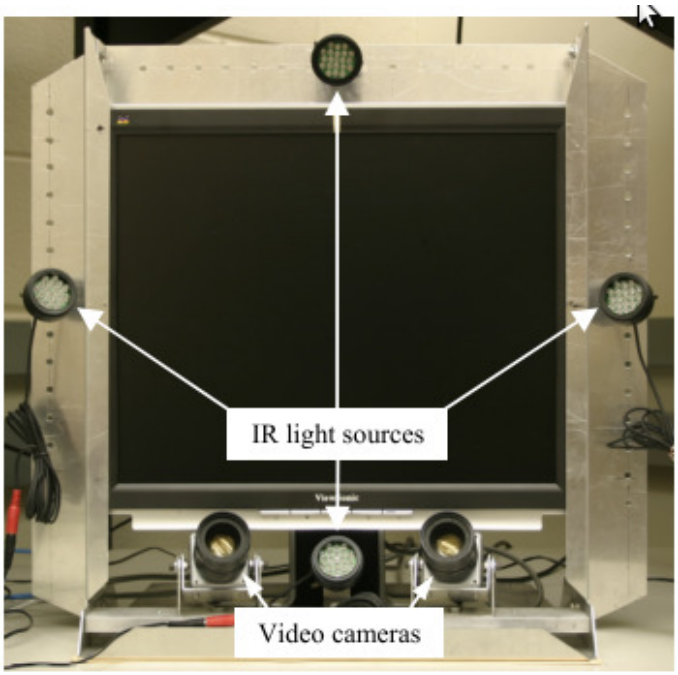



**Figure 1.6:** One of the multi device gaze tracking system which is proposed in [3].

several advantages of Kinect. First, this camera provides intensity and depth data in real-time. The gaze estimation methods which employ 2D images can be used for gaze tracking and the head pose estimation methods which employs 3D data can be used for head pose estimation. As a result, a unified system using one piece of hardware can be achieved. Although, there has been a lot of research on 3D head pose estimation, accurate 3D head pose estimation in real-time needs to be investigated more. The majority of research on head pose estimation has been performed on databases containing few head position variations and limited work on head pose estimation has been conducted on data containing large head movements. A brief overview of the approaches proposed for 3D head pose estimation in the past few years can be found in [9].

## 1.4 Objectives and Overall Project Description

The objective of this research is the development of methods for appearance-based gaze tracking. Appearance of eyes is the way of a user eyes' looks to other humans. The appearance-based gaze tracking is the method which is based on imaging users' faces by a camera and then mapping eye regions on the image to the locations of their focus of attention. However, the accuracy of appearance-based methods degrades when a user

moves away from the initial calibration position. The user's head pose can be used to compensate eye appearance distortion for more accurate estimation of the focus of attention. Therefore, another objective of this research is to estimate the users' head pose with respect to the camera in real-time.

First objective is to develop a method which allows the user's head pose parameter estimation from 3D data of a RGB-D camera (Kinect). Real-time head pose estimation methods are investigated. In particular, methods that employ 2D images and 3D depth data are studied. Random regression forests are studied together with other regression methods for parameter estimation. Random regression forests provide fast computation which is crucial for real-time performance. The objective is to combine random forests and other regression methods in order to achieve a more accurate head parameter calculation. Another objective is to extend the random forest methods to employ 3D data and 2D intensity images from the Kinect camera for accuracy and robustness.

Second objective is to develop an appearance-based gaze estimation method which allows gaze estimates when users moves with respect to the camera. This can be achieved by using the proposed head pose estimation methods in conjunction with the gaze estimation methods. First, methods should be developed to estimate gaze in frontal pose. Then these methods should provide gaze estimates when users move away from initial calibration point. Furthermore, incremental learning methods should be investigated and a new incremental based technique should be proposed to achieve appearance-based gaze estimation. The motivation behind incremental appearance-based gaze estimation is as follows. Both training and test samples can be used during the model generation for obtaining more accurate gaze parameter estimates. Users positions with respect to camera or alignment of the eye patches during cropping leads the models to provide less accurate estimates. The incremental model update is expected to reduce these problems. The incremental model update during online gaze estimation might also be used to adapt to different users online without a recalibration process. Furthermore, the methods proposed above should be extended in order to achieve gaze tracking under free head pose. In particular, head pose estimates should be used to compensate for head pose variation during gaze estimation.

## 1.5 Contributions

The contributions of the thesis can be summarized as follows.

Recently, low cost RGB-D cameras provide RGB images and 3D depth data. Since these sensors provide RGB images and depth, the fusion of RGB and depth data can be employed for accurate head pose estimation. It is well know that random regression forests provide accurate estimates and fast computation time. Therefore, random forests can further be developed for head pose estimation. In this thesis, a method which is based on a random forest and tensor models is proposed. In particular, a random forest is further extended in three ways: (i) by using tensor-based regression at each leaf node, (ii) by fusing depth and intensity data using tensor regression at each leaf node and (iii) fusing RGB and depth data using random forest. As a result, a new method based on multi-modal random forests and tensor models is proposed for more accurate and robust head pose estimation. The random forests allow modeling large head pose while tensor models provide accurate results.

One of the problems of appearance-based methods is that their accuracy degrades when users move from a initial calibration position to a new position. Two novel methods are proposed to addressed this problem. First, the proposed methods for head pose estimation are used in conjunction with the proposed appearance-based gaze estimation method. The distortion due to head pose on eye images is eliminated using perspective projection. After, the gaze estimates are further improved by eliminating bias between gaze estimates and the actual values of gaze parameters. The bias are learned using a regression model. Second, a novel method which allows gaze direction parameter learning online is introduced. The motivation of online learning is to update models to deal with new eye image appearances. The novel proposed method is based on generating a Gaussian Mixture model (GMM) of the training data and online Support Vector Regression (OSVR). The training data is also modeled using eigenspace decomposition for computational purposes. The GMM model enables the detection of any changes on eye image appearances and OSVR models are updated for these new appearances online. In this way, a recalibration process is not required when a user moves from an initial calibration position to a new position.

## 1.6 Thesis Organization

The following chapters of the thesis can be organized as follows.

In Chapter 2, a review of related work regarding head pose and gaze estimation is given. First, 3D and the appearance-based gaze estimation approach are presented. The

theory of gaze estimation is also described by giving a graphical representation of the eyeball structure and screen coordinate system. After, conclusions are reported related to the strengths and weakness of the 3D and appearance-based approaches. One of the weaknesses of the appearance-based methods is that their accuracy degrades when a user moves from the initial calibration point. This problem can be addressed by making use of head orientation and position parameters when mapping 2D eye images to gaze points. Since, head pose parameters are necessary when estimating gaze under free head pose, the head pose estimation approaches are also reviewed and their strengths and weaknesses are reported. In particular, the methods that employ 3D depth data, 2D images and combinations of these two are discussed.

In Chapter 3, an overview of the proposed gaze estimation method which is based on the proposed head pose estimation methods is described. First, an overview of the proposed online gaze estimation system is introduced. Second, random forests are presented. Since the proposed head pose estimation methods are based on the combination of a random forest and tensor regressors, the strengths of these methods are described. Tensor learning which is an extension of Support Vector Regression (SVR), is also explained together with SVR. Second, more details of the proposed online gaze estimation system which is based on a Gaussian Mixture Model (GMM) and online Support Vector Regression OSVR are given. After, the detailed explanation of GMM and OSVR are provided. The main advantages and disadvantages of these methods are discussed. In particular, the main strengths of OSVR over SVR are discussed. Furthermore, eignspace representation of data is also explained.

In Chapter 4, a proposed method, called multi modal random forest based tensor regression, for real-time head pose estimation using both depth and intensity data is introduced. We address the problem of head pose estimation by extending previous methods on random forest based head pose estimation in three ways: i) by using tensor-based regression at each leaf node, ii) by fusing depth and intensity data using tensor regression at each leaf node, and iii) by fusing depth and intensity data using random forest framework. The proposed method will be compared to current state of the art approaches in terms of accuracy.

In Chapter 5, two proposed methodologies for appearance based gaze estimation under free head pose are introduced. In the first approach, perspective projection is employed using head pose parameters for head orientation compensation. Translation compensation is also employed by learning gaze direction bias. Several regression models are used for mapping eye images to gaze direction parameters and translation bias.

Experimental results are provided for performance comparison of proposed methods and other methods. In the second approach, a novel method which allows gaze direction parameter learning online is introduced. The proposed method allows updating models online to deal with new eye image appearances. The novel proposed method which is based on generating Gaussian Mixture model (GMM) of the training data and online Support Vector Regression (OSVR) is introduced. Experimental results are provided for performance comparison of proposed methods and other methods.

In Chapter 6, the limitations of the previous gaze estimation methods were restated. The proposed gaze estimation methods for addressing these limitations were summarized. Since the proposed gaze estimation methods were combined with the proposed head pose estimation methods, the proposed head pose estimation methods were also summarized. In the conclusion section, the performances of proposed gaze estimation methods have been assessed by considering the experimental result which were reported in chapter 5. Furthermore, the performance comparisons of proposed gaze estimation methods and the previous proposed gaze estimation methods were reported. Similarly, the performances of the proposed head pose estimation methods have been assessed. Moreover, the performance comparisons of the proposed head pose estimation methods and the previous proposed head pose estimation methods have been given. Finally, possible future of the developed gaze estimation methods have been demonstrated. Similarly, possible future of the developed head pose estimation methods have been demonstrated.

## 1.7 Publications

- S. Kaymak, I. Patras. "Multimodal random forest based tensor regression", IET Computer Vision, vol. 8, no.6, pages. 650-657, 2014
- S. Kaymak, I. Patras. "Exploiting depth and intensity information for head pose estimation with random forests and tensor models", Proc. Asian Conf. on Computer Vision Workshops, page 160-170, 2012.

# Chapter 2

# Related Work

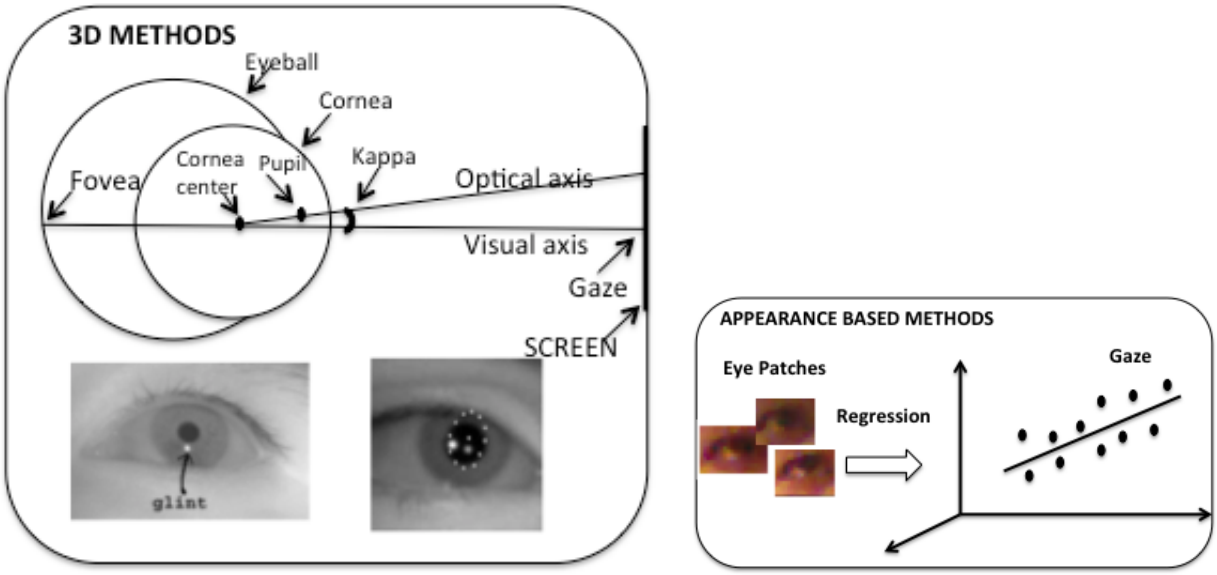
## 2.1 Introduction

In this chapter, we review the state-of-the-art related to head pose estimation, and gaze estimation will be described. First, the methods which are based on 3D and appearance based eye gaze estimation will be described and discussed. Second, the methods which are based on 2D image and 3D data for head pose estimation will be discussed. Since appearance based gaze estimation requires detection of eye regions on the face, 2D image and 3D data based facial feature detection methods will also be reviewed.

## 2.2 Gaze Estimation

Gaze estimation methods can be divided into two groups: 3D gaze estimation methods and appearance based methods. 3D gaze estimation methods are based on localizing 3D eye features such as the corneal center, and the pupil center for eye gaze direction calculations. These methods allow better accuracy under free head pose than appearance-based methods. Appearance based methods consider the eye region as a high dimensional feature vector and developed functions that allow mapping from this features to gaze points. The main drawback of the appearance based gaze tracking is the head pose variation. The accuracy of gaze estimates are degraded when the user's head pose in front of a camera change. The appearance mapping under head pose requires the head orientation in 3D space in order to eliminate the head pose variation. First, the description of the gaze point in 3D space will be described by introducing the eye ball structure in the following

section. Then, a brief overview of the 3D gaze estimation methods will be discussed. Figure 2.1 shows a comparison of 3D and appearance based gaze tracking [30].



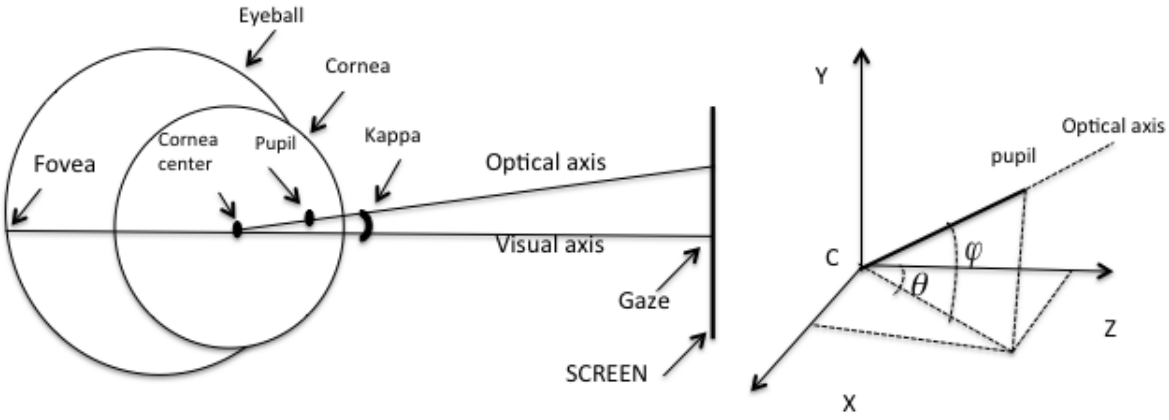

(a) (b)

**Figure 2.1:** Comparision of 3D and appearance based gaze tracking. (a) 3D based methods, (b) Appearance based methods.

### 2.2.1 Introduction

The structure of the eyeball, can be seen in the Figure 2.2(a). This structure consists of two spheres: eyeball and cornea. The optical axis is defined as a line passing through the pupil and the center of the cornea. The visual axis passes through the corneal center and the center of the fovea. The gaze point is defined as the intersection point between the visual axis and the screen. The optical axis direction can be used to obtain the visual axis using an angle (kappa). The value of kappa changes from person to person. This angle is determined through the calibration process.

The orientation of the optic axis can be seen in Figure 2.2(b). The direction of this axis is defined by horizontal, $\theta_{eye}$, and vertical $\varphi_{eye}$ angles. The unit vector of the optic




(a) (b)

**Figure 2.2:** Description of visual axis in the world coordinate system, [4], (a) the eyeball structure, (b) the optical axis angles.

axis is defined as [4, 31, 32]

$$\mathbf{v}_o = \begin{pmatrix} cos(\varphi)sin(\theta) \\ sin(\varphi) \\ cos(\varphi)cos(\theta) \end{pmatrix} \tag{2.1}$$

The visual axis can be estimated from the optic axis by adding $\kappa = (\alpha, \beta)$ to the optic axis

$$\mathbf{v}_g = \begin{pmatrix} cos(\varphi + \beta)sin(\theta + \alpha) \\ sin(\varphi + \beta) \\ cos(\varphi + \beta)cos(\theta + \alpha) \end{pmatrix} \tag{2.2}$$

The kappa ($\kappa$) angle can be seen in Figure 2.2(a). The unit vector in the direction of visual axis with respect to World Coordinate System (WCS) is represented as

$$\mathbf{v}(\theta, \varphi) = Rv_g(\theta, \varphi) \tag{2.3}$$

where $\mathbf{R}$ is the rotation matrix from eye coordinate system to WCS. This matrix is calculated using the orientation of the optic axis of the eye and the Listing's Law. The

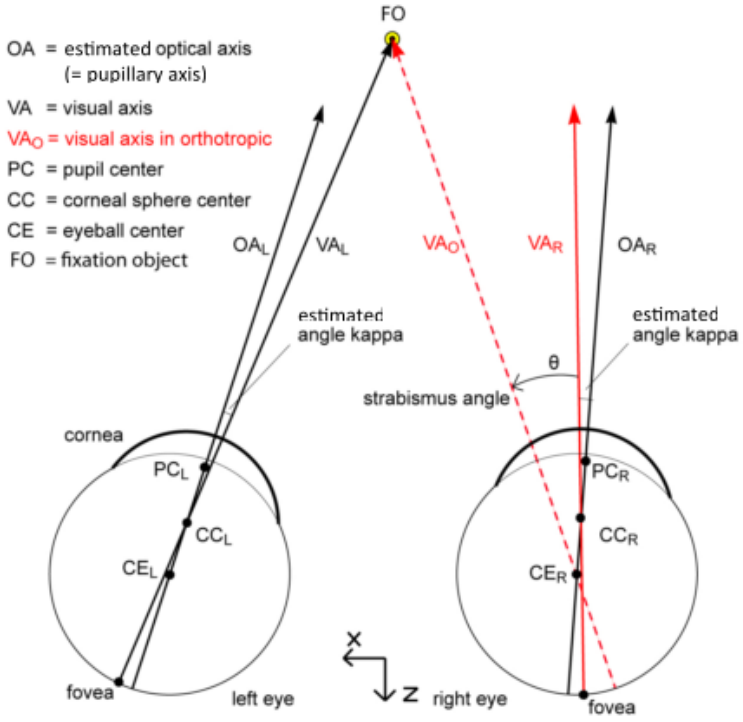



**Figure 2.3:** The desctiption of the strabismus angle. [5]

gaze point can be calculated by

$$\mathbf{\Psi}(\theta,\varphi) = \mathbf{c} + k(\theta,\varphi)v(\theta,\varphi) = \mathbf{c} + k(\theta,\varphi)\mathbf{R}\mathbf{v}_g(\theta,\varphi) \tag{2.4}$$

where $\mathbf{c}$ is the cornea center and $\mathbf{o}$ is the optic axis. k is a constant value defined by the intersection of the visual axis with the observation surface. If plane (display surface) is defined by $\{\mathbf{x} \backslash \mathbf{n} \cdot \mathbf{x} + h = 0\}$, then $k(\theta,\varphi)$ is given by

$$k(\theta,\varphi) = -\frac{h + \mathbf{n} \cdot \mathbf{c}}{\mathbf{n} \cdot \mathbf{v}(\theta,\varphi)} \tag{2.5}$$

where $\mathbf{n}$ is the normal to the plane surface.

### 2.2.2 Strabismus Angle

As stated in Section 1.1, strabismus angle is a condition where a person can not look at a fix point with both eyes, [5]. In [5], a method is proposed to estimate strabismus angle of a person. Figure 2.3 demonstrates the strabismus angle of a subject. As it can be

seen in the figure, the visual axis of the left eye intersects with the fix point; however, the right eye's visual axis does not intersect with this point. The angle between visual axis and visual axis in orthotropic is called strabismus angle.

### 2.2.3 3D Gaze Estimation Methods

A number of 3D gaze estimation methods have been proposed. Several methods have been proposed to estimate eye gaze using stereo cameras [33], [34] [32], and [35]. A method [32] was also proposed to estimate the eye gaze using multiple light sources and a single camera. Chen and Ji [4] presented an incremental probabilistic 3D gaze estimation method which allowed free head movement without explicit calibration.

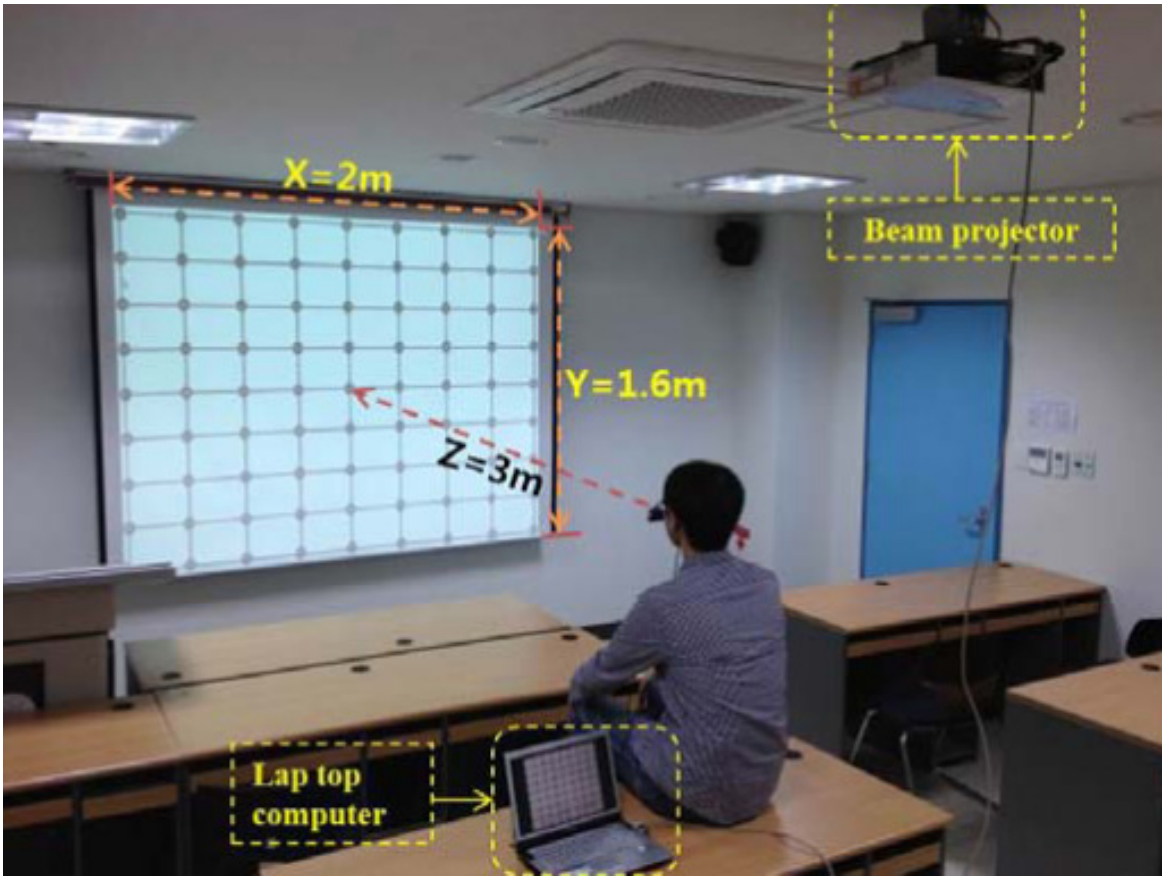



**Figure 2.4:** One of the experimental set-up using a large screen, [6].

Recently proposed gaze tracking methods allows gaze point estimation on large screens (3.2 x 2.4 m) [6, 36, 37]. These methods are based on wearable devices such

as a helmet or a pair of glasses. The main advantage of these methods is that the head pose is not required to be know or estimated. In [6], the user's eye image is captured by a camera using NIR illuminator. The pupil center is detected in a captured image. Then the user is instructed to look at the four corners of the screen. The corresponding eight feature values of four pupil centers are used as inputs to a multilayer perceptron in order to generate five additional points. Finally, first order polynomials, [38, 39], are used to map pupil centers to screen coordinates.

### 2.2.4 Appearance-based Gaze Estimation Methods

Baluja and Pomerleau [40] proposed a neural network method to obtain a mapping function from eye appearance to gaze points. 2000 training samples were used to obtain this function. Xu et al. [41] proposed a similar approach. Tan et al. [26] proposed to represent eye appearance using local linearity of the eye appearance. These representations provided the mapping from features to gaze point. Williams et al. [42] proposed a semi-supervised method based on Gaussian Process Regression in order to obtain a mapping function. In both approaches, the rotation and translation values were calculated using a computer vision-based head tracker.

The limitation of the above approaches was that they were developed using eye appearance under a fix head pose. There is limited work reported for gaze estimation in the presence of head pose. Sugano et al. [43] proposed an incremental learning method for unconstrained gaze estimation. Their approach was based on non-linear dimensionality reduction by locally linear embedding [44]. Eye appearances were extracted and represented with weight vectors that were clustered according to head pose. The number of clusters were created while subjects were clicking points on the screen. In the gaze estimation stage, head pose was estimated and given the head pose parameters, the corresponding cluster was selected. The selected cluster provided the estimated gaze vector. Lu et al. [45] proposed a head pose-free approach for appearance-based estimation. Their method was based on three stages of gaze estimation. First, gaze was estimated using an interpolation approach by assuming a fix head pose. Second, an estimated 3D norm of gaze vector was rotated according to estimated head pose. Finally, the distortion of appearance was compensated using a transformation matrix which was learned from regression. Recently proposed gaze tracking methods, [30, 46] proposed to estimate gaze direction by generating a 3D face of a user. In [46], eight images were captured from eight cameras and a 3D face was formed. Using a 3D face, synthetic images

are generated and used as input to a random regression forest for training. Synthetic image generation creates a training set for a regression model and avoids long calibration process. Similarly, authors, [30], propose to use a Kinect camera in order to make use of depth image of a person. The intensity image is mapped onto the depth data. In this way, the distortion of the appearance of the eye regions are eliminated by re-rendering the face to frontal pose. In their study, the geometric models are formulated in a probabilistic framework. In this way, they both introduce synthetic image generation and estimate gaze direction under free head pose.

**Figure 2.5:** Images and fixations of seven subjects [7].

### 2.2.5 Gaze Estimation without Personal Calibration

Recently, gaze estimation methods which do not require calibration have been proposed. Model and Eizenman [31] obtained estimated eye gaze direction parameters using pyramid observation surface or large screen. Their proposed method depends on the availability of the large screen or pyramid surface.

The saliency of the images for predicting human attention was first proposed by Koch and Ullman [47]. After Koch's study, a number of approaches, [48–51], were introduced to exploit the saliency of images for predicting human attention. Saliency map generation can be divided in two main categories: those that generate saliency maps using a mathematical model and those that learn saliency maps from datasets of

**Figure 2.6:** Judd study showed that subjects gaze at specific parts of faces in images. These parts includes eye, nose and mouth regions [8].

eye gaze data. Bruce and Tsotsos, [52], introduced an information theoretic approach. Avaramham and Lindenbaum proposed a stochastic model to estimate most saliency maps . Itti et al. [53] present a method to combine low level features, color, intensity, and orientation, in order to generate saliency maps. Cerf et. al. [7] presented a method which makes use of low level features and a face detection method to obtain more accurate saliency maps than the Itti method. Saliency map modelling using Graph-based saliency map with Viola-Jones face detector provided better saliency maps. Their dataset consists of 250 images and gaze points of seven subjects. Figure 2.5 shows several images and gaze points. Kienzel et al. [51] proposed to learn saliency maps using gaze dataset. Patches were extracted on gaze points and used as salient patches. Patches were also extracted around gaze points and used as non-salient patches. Low level features were calculated using these patches in the mapping process. Their eye movement dataset consisted of 200 grayscale natural scene images. A similar method was proposed by Judd et. al. [8]. The saliency maps are learned from a larger dataset which consisted of 1003 images with associated eye gaze data from fifteen users. In addition to low level features, low level, mid-level and high-level features were learned for saliency map generation. Figure 2.6 shows several images and gaze points. The works from Chen and Li [4], and Sugano et. al., [54] describe eye gaze estimators from saliency maps. The saliency of the images has been used for automatic generation of models that enables mapping from eye features to eye gaze points on the screen. The method proposed by [4] is an incremental probabilistic

3D gaze estimation under natural head pose without active calibration. Their proposed method is based on the combination of 3D eye gaze and the saliency map [48]. The saliency map provided estimated eye gaze points and the parameters of 3D eye gaze direction was learned using these estimated points. In contrast, Sugana proposed an appearance based gaze estimation without active calibration. Subjects watched a video. Estimated gaze points were retrieved and eye images were mapped to gaze points using Gaussian Process Regression. The conclusions of this section are given in Section 2.2.6.

### 2.2.6 Discussions and Conclusions

In this section, the-state-of-the-art 3D and appearance-based gaze estimation methods were reviewed. First, the proposed 3d calibrated and uncalibrated methods were reviewed. 3D calibrated methods were based on displaying several points on the screen and instructing a user to look at these points. About nine calibration points were used to estimate a user's gaze direction parameters. 3D uncalibrated gaze estimation methods [4] were based on displaying several images on the screen and then learning a user gaze direction parameters. Saliency maps of displayed images were calculated and possible locations of gaze points were obtained for calibration. Second, the calibrated and uncalibrated appearance-based methods were also proposed. The appearance-based calibrated methods [26, 45] were also based on displaying several points on the screen and instruction a user to look at these points. Between nine and thirty-three calibration points were used to estimate a user's gaze direction parameters. The number of calibration points used for the appearance-based methods were higher than the number of calibration points used for 3d methods. Uncalibrated appearance-based methods [54] were performed by displaying several images on the screen and then instructing a user to watch these images. The calibration of process was performed by calculating saliency maps of displayed images and calculating probable focus of attention points. Then eye image features were map to these locations. The-state-of-the-art showed the uncalibrated appearance-based methods reqired more images to be displayed on the screen to achied the same performance as the uncalibrated 3d methods. Although calibration is less boring and more enjoyable for the 3d and appearance based uncelebrated methods, the calibration time became longer.

## 2.3 Real-time Head Pose Estimation

Formally, the head pose estimation is the problem of estimating the head orientation angles, usually in the camera coordinate system [9]. These orientation angles of an average adult male in the forward and backward direction range from -60.5 to 69.6 degrees, in right and left bending of the neck range from -40.9 to 36.3 degrees, and in horizontal axis rotation ranges from -79.8 to 75.3 degrees. The head rotation has three DOF, sometimes called pitch, yaw and roll as shown in Figure 2.7. Head pose is directly related with the gaze direction [9, 55] and can provide coarse gaze estimates during occlusions of the eyes, e.g. in the presence of eye glasses.

Head pose estimation approaches can be divided in three main categories: those that employ 2D images, those that employ 3D depth data and those that combine 2D images and 3D depth data.

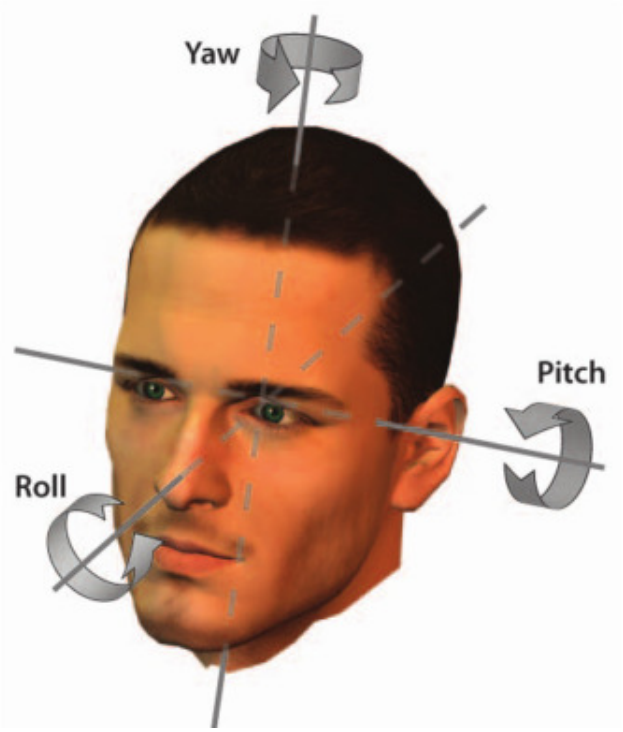



**Figure 2.7:** A human head pose can be described by three rotation angle which are pitch, yaw and roll [9].

### 2.3.1 2D Data based Methods for Head Pose Estimation

Head pose estimation approaches that employ 2D images can be further divided into two groups: 2D appearance based methods and 2D feature based methods. The 2D appearance based techniques analyse the entire head region. Osadchy et al. [56] proposed a Convolutional Neural Network based large head pose estimation system which allowed mapping face images to head pose parameters and achieved a near real-time performance (5fps). The modelling of facial regions was also used for tracking using statistical techniques such as Active Appearance Models (AAMs) [57], multi-view AAMs [58], and 3D Morphable Models [59], [60] and Constrained Local Models [61]. In [62], tensor-based regression is compared to Support Vector Regression specifically for the problem of head pose estimation in the IDIAP [63], Boston University dataset [64] and Pointing4 , [65], datasets. The reported results on these datasets showed that tensor based regression (higher rank Support Tensor Regression ) performed better than other regression algorithms for the head pose estimation. 2D feature based methods are based on facial feature detection for the head orientation calculation. Vatahska et al. [66] proposed detecting facial features and estimating head orientation using the detected locations of the points in three stages. First, the head pose was classified as frontal, left and right profile using a face detector. Then the facial features were detected by training AdaBoost classifiers with Haar-like features. Finally, the locations of detected features were mapped to head orientation parameters using neural networks. Whitehill et al. [67] proposed a method in which the orientation of a head was calculated using locations of the node tip and both eyes.

### 2.3.2 3D Data based Methods for Head Pose Estimation

The second category of head pose estimation systems uses 3D depth data. The system proposed by Breitenstein et al. [68] allowed head pose estimation from depth data in real time. The real time performance has been achieved using a GPU. Large 3D face depth data in different head poses was stored into the GPU memory and a unknown depth data was recognized by comparing with the stored data. Real-time head pose estimation techniques also employed Random Regression Forests, [12], [18]. In [12] the authors generated large 3D synthetic faces and trained Random Forests for continuous head pose estimation. They also extended this technique and achieved joint classification and regression for head pose estimation in [18]. This system allowed extracting patches from the upper body region of a person from depth data and only patches which belonged

**Figure 2.8:** Localized facial features on test 2D images [10].

to the head region were used to estimate the head pose in real time. The estimation was performed using low resolution data captured by Microsoft Kinect Camera.

### 2.3.3 2D and 3D Data based Methods for Head Pose Estimation

The third category involves the combination of 2D and 3D data. Seemann et al. [69] presented a head pose estimation system based on neural networks. Grayscale and depth data were used as inputs to the neural networks to calculate the head pose. Morency et al. [70] calculated a prior model of the face using intensity and depth images. This model was used to calculate the absolute difference in pose for each new image.

## 2.4 Real-time Facial Feature Localization

Facial feature localization can be divided in three main categories: those that employ 2D images and those that employ 3D depth data.

### 2.4.1 2D Data based Methods for Facial Feature Localization

Dantone et al. [10] proposed conditional random regression forests in order to localize facial parts on 2D images. In their study, head pose variation in the z-axis was represented with 5 subsets and a forest was trained for each subset. In the testing, the forest was

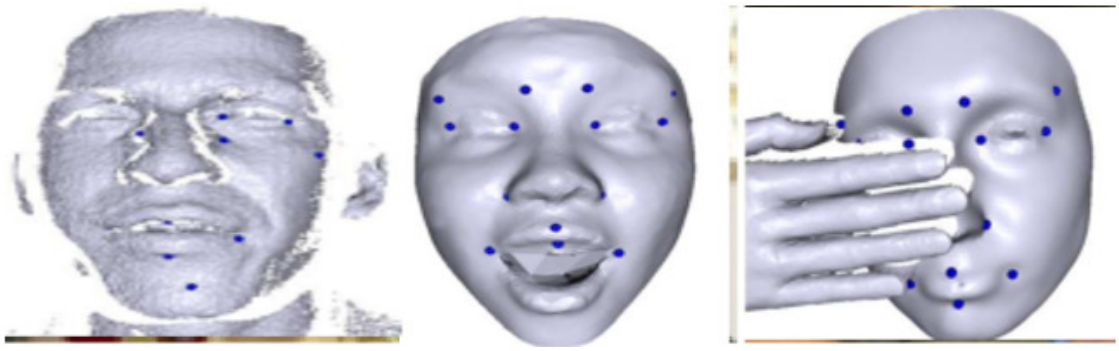


**Figure 2.9:** Localized facial features on test 3D depth data [11].

determined by localized head pose and then used to detect facial parts. Facial features locations can be seen in Figure 2.8.

### 2.4.2 3D Data based Methods for Facial Feature Localization

Available methods for facial feature localization in real-time can be divided into two categories: those that employ entire face information and local information. Wiese et. al. [71] proposed a method in order to track face motion in real-time using person specific templates. They extended their work to estimate head pose and facial deformation by employing RGB image and depth data captured by Kinect data [72]. A number of approaches also exists for localization of feature points using local information. Mehryar et. al [73] achieved facial point localization by clustering surface curvature values which were calculated on 3D depth data. Wang et al. [74] presented a method which allowed the facial point detection on 3D data and 2D image. The point signatures and Gabor filters were calculated and used to detect the landmark locations. Fanelli et al. [11] presented a random forest based facial feature localization in real-time. Their approach was based on voting. Fixed size patches were extracted from face regions and these extracted patches voted for facial point locations. Their approach was robust in terms of large head pose, partial occlusions, and noisy depth data. Facial feature locations can be seen in Figure 2.9.

## 2.5 Discussion and Conclusion

The related work of gaze estimation methods was described in this chapter. Two main approaches, 3D and appearance-based, have been explained. After considering all related work, several conclusions can be drawn. First, the appearance-based gaze estimation methods were combined with head pose estimation methods. The 2D eye image features change while the head moves away from the calibration position. Changes of feature values degrades the gaze estimates. This degradation was eliminated using head pose estimates. Second, 3D and appearance-based uncalibrated gaze estimation methods were introduced to avoid calibration process. However, these uncalibrated methods required longer time to allow for a similar performance as the calibrated methods. In conclusion, two new methods are proposed to address the existing problems of the appearance-based methods. First, a new head pose estimation method will be introduced and combined with a new appearance-based gaze estimation method in a unified system. The aim is to achieve accurate gaze estimation which gives a better performance than the previous methods. Second, an online gaze estimation method based on the proposed head pose estimation method is proposed in chapter 5 to achieve gaze estimates with less calibration time and more accurate gaze estimation.

The related work of head pose estimation methods was described in this chapter. 2D, 3D and 2D and 3D methos have been explained. After considering all related work, several conclusions can be drawn. There have been a number of approaches to the head pose estimation. However there are several problems which need to be addressed. The majority of the previous work reported the accuracy of head pose estimation only without indicating computational complexity for real-time implementations. Accuracy has been improved however the improvements are still not adequate for some applications. In addition, few studies reported the combination of several cameras such as color and depth camera. Furthermore, the majority of the studies perform head pose estimation based on images where the face is near frontal. Head pose estimation from profile images needs to be addressed in terms of accuracy. Finally, the majority of algorithms have been developed for sequential implementation on Central Processing Units such as Intel CPU. Novel head pose algorithms which allow combination of several methods (appearance and model based) can be proposed. However, sequential implementation of these algorithms might be problematic for real-time implementation. On the other hand, two recent works by Breitenstein et al. [68] and Fanelli et al. [12, 18] allowed head pose estimation under large head variation in read-time. First, the work [68] makes use of

parallel processing units of a GPU. A number of face templates are stored into the GPUs memory and then these face templetes are rendered according to the nose location in captured range data. This method is sensitive to occlusions. This method can also not to be used for several applications since the GPUs requires high power. Second, Fanelli et. al. [12, 18], proposes a robust head pose estimation method which allows real time performance under large head pose variation and occlusion. This method is based on Random forests which allows large head pose estimation and fast computations without the need of a GPUs. In conclusion, Random forests allow modelling large head pose in real-time. They also allow head pose parameter estimation under occlusion. Therefore, these methods will further be extended and improved to obtain more accurate head pose estimates. In chapter 3, the description of Random forests will be given. Combination of random forests with Tensor models will be introduced in Chapter 4.

# Chapter 3

# Background Theory

## 3.1 Introduction

A review of the previous gaze estimation methods is given in the previous chapter. The existing limitations of these gaze estimation methods were also discussed. Since the appearance based methods are combined with the head pose estimation methods for eliminating distortion on eye images for accuracy, the head pose estimation methods are also reviewed. A number of methods are described and compared. In this chapter, the proposed methods for addressing these problems are overviewed

One of the limitations of appearance-based gaze estimation methods is that their accuracy degrades when a user moves from the initial calibration position. This results in different eye appearances of a user and regression models can not deal with this. An online gaze estimation method is proposed to address this problem. This system is also combined with the proposed head pose estimation method which is based on random forests and tensor regressors. The proposed head pose estimation method is used to eliminate the distortion on eye images due to head pose variation. An overview of this system which is based on a Gaussian Mixture Model (GMM) and online Support Vector Regression (OSVR) is as follows. The GMM model allows selecting test eye patches online for OSVR mode update. The selection of eye patches online is achieved by modeling the training eye patches using the GMM. The GMM consists of a number of Gaussian models which allow accurate modeling of the training eye patches. After OSVR models are updated using selected eye patches. Fast computation is also achieved by representing the eye patches in eignspace. Eignspace representation allows modeling eye patch by GMM and rapid updating of the OSVR models. The main advantage of

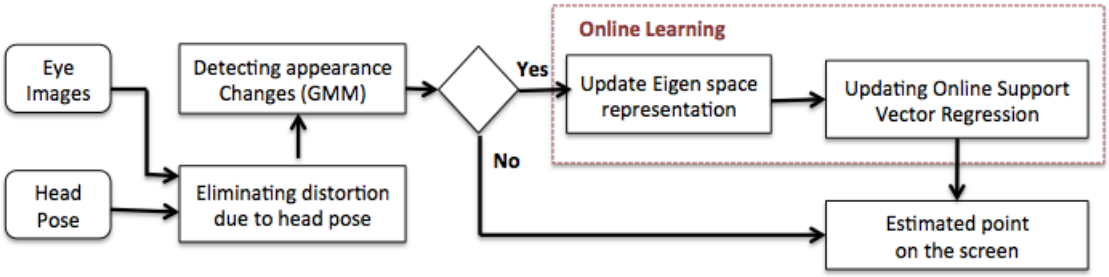



**Figure 3.1:** The proposed online learning method.

OSVR is that training is performed by adding or removing a sample from the training set without retraining the models from scratch for each new sample. This is very a crucial strength in our proposed method because updating models is achieved in real-time. OSVR is different than SVR because SVR requires models to be trained from the beginning for every new eye patch. This causes more computation time and SVR might not lead to real time computation. Figure 3.1 shows the proposed online learning method. Table 3.1 provides the summery of functions of each method which are used in the proposed online gaze parameter learning system.

The proposed head pose estimation method which is stated above is based on random forests and Tensor regression. The previous work shows that random forests have several advantages over other regression methods. First, the method allows easy implementation and fast computation time. Second, random forests allow for the combining of classification and regression methods in a unified method. Combining classification and regression methods in a unified system allows detecting face regions on depth data using the classification method and then mapping these regions to head pose parameters using the regression method. Furthermore, since random forests are based on extracting fixed size patches on the intensity images and depth data, the occluded face regions can be discarded and unoccluded patches are used for head pose estimation. Therefore, a new method which is based on random forests and Tensor learning for regression is proposed in this thesis. This method has several advantages over classic random forests. First, tensor regressors are trained to map patch data to head pose parameters for more accurate head pose parameters. Second, the proposed method employs both intensity and depth data for more a robust and accurate head pose parameter estimation. Since the proposed method is based on random forests and tensor models, ransom forests and tensor regression for learning is described together with Support Vector Regression. Figure 3.1 shows the combined head pose and appearance-based gaze estimation methods.

| METHOD | DESCRIPTION |
|---|---|
| **PCA** | Allows low dimensional representations. This has two advantages. First, modeling the eye images using a Gaussian Model resulted in less computation time. Second, updating online Support Vector Regression models online resulted in less computation time. |
| **GMM** | Allows accurate data modeling. It also provides supervision during updating online Support Vector Regression models. |
| **OSVR** | Allows fast mapping from projection vectors to gaze direction parameters online. |
| **Random forests based tensor regression** | Allows head pose parameter estimates in real-time from intensity and depth data captured from RGB-D camera. |

**Table 3.1:** Descriptions of proposed methods for Online adaptive gaze estimation.

The remainder of this chapter is as follows. Random forests are described in Section 3.2. The related work related to random forests is given in Section 3.3. Then offline and online SVR methods are discussed in Sections 3.4 and 3.5 respectively. Finally, the principal component imagery is described in Section 3.6 and a Gaussian mixture model is described in Section 3.7.

## 3.2 Random Forests

Decision trees, [75, 76] can be used for classification and regression. These trees allow grouping input data into clusters and then models are generated to map data in clusters to discrete or continuous outputs. A tree consists of non-leaf nodes and leaf nodes. Non-leaf nodes contain binary tests which guides the input data at parent node to the left or right child node. These binary tests are determined during tree training. Then, the splitting of data at non-leaf nodes continues until a stopping criteria is met. Each leaf node then contains part of the data that is modeled by predictors to give the mapping to the output parameters.

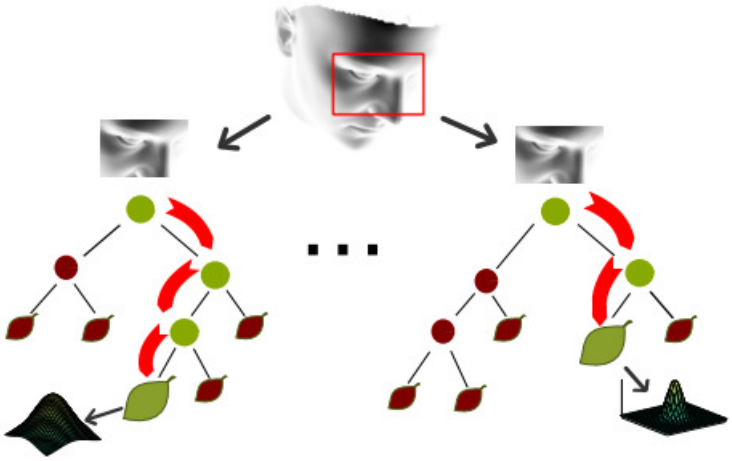


**Figure 3.2:** Regression forest for head pose estimation. Input samples travel through the forest according the stored information at each non leaf node. Multivariate distribution of models at leaf nodes provide estimates of head pose parameters [12]

Decision trees were further extended by Breiman, [75], to a collation of trees called random forests. These ensembles of trees are trained separately using data samples that are randomly extracted from the training set. The main advantage of random forests is that overfitting is reduced when compared to a single tree.

One of the examples of random forest related to head pose estimation can be seen in Figure 3.2. Random patches are extracted from the face region of a depth data. These patches are passed through the forest. Patches are guided at non-leaf nodes until they reach leaf nodes. The predictors model head orientations at lead nodes. An overview of a random forest training can be described as follows. A random forest, a collation of trees, can be denoted by, $T = \{T_t\}$, where $T_t$ is a tree in the forest. Trees in the forest are trained using randomly extracted patches, $P = \{P_i\}$, from the images. The appearance of the patch can also be denoted by $I_i$. A set of patches is passed to a tree starting from the root node. These patches pass trough the tree and split at each non-leaf node according to binary test, $\phi(I) \rightarrow \{0, 1\}$. The binary tests at non-leaf nodes are calculated by generating a pool of tests and selecting one which maximizes the information gain. The information gain is defined by

$$\phi = \arg\max_{\phi} IG(\phi) \tag{3.1}$$

$$IG(\phi) = H(P) - \sum_{i \in L,R} w_i H(P_i)(\phi) \tag{3.2}$$

where $w_i = |P_i(\phi)|/|P|$ is the ratio of patches sent to each child node and $H(P)$ is the entropy of the clusters' labels. $H(P)$ is defined according to classification and regression tasks. The splitting process continues until the patches reaches the leaf nodes. Leaf nodes are created when the maximum tree depth is reached or the number of training samples are less than a minimum number training samples.

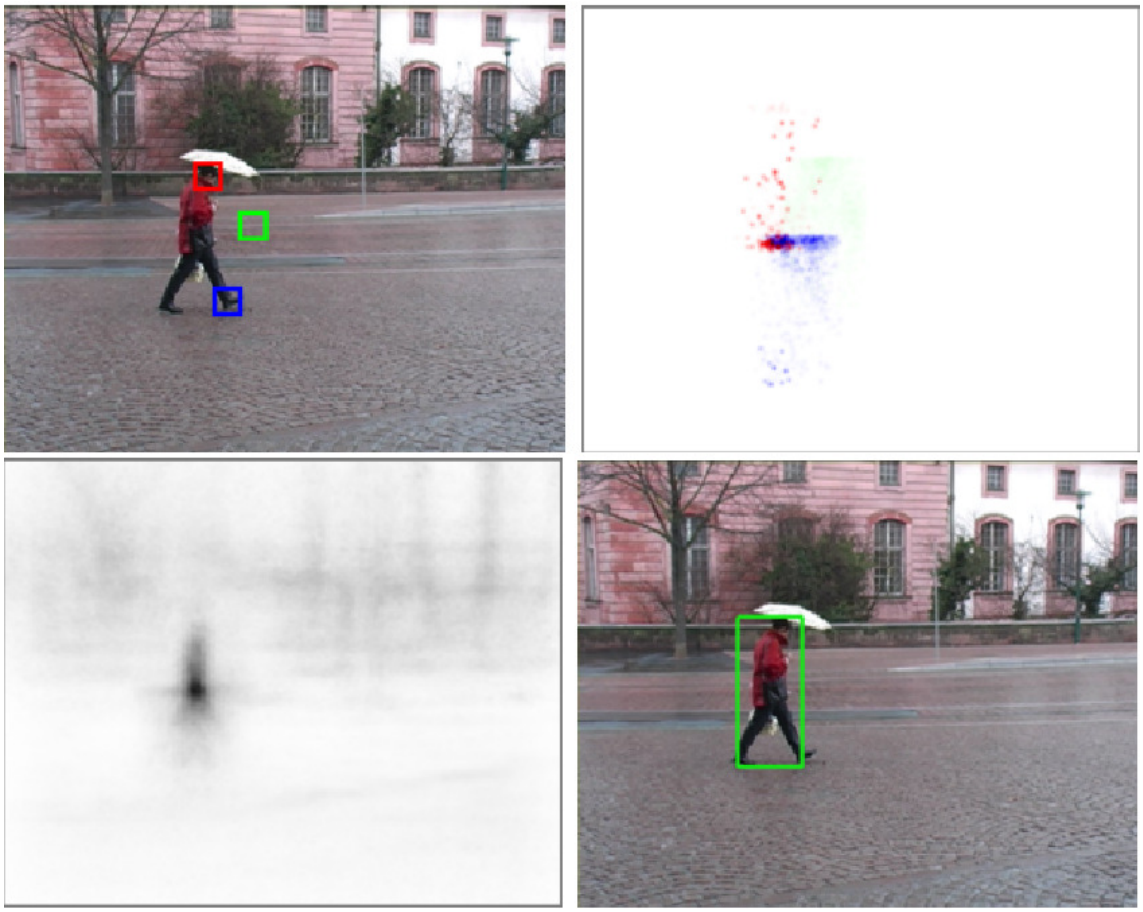


**Figure 3.3:** Description of the patch voting in Hough space, a) three example patches. b) vote information obtained from Hough forests. c) votes are aggregated in Hough space. d) the pedestrian detected in this image, [13].

## 3.3 Related Works based on Random Forests

Random forests, [75, 77] allow fast computations for real time applications. Random forests have become a popular method in computer vision given their capability to handle large training datasets, high generalization power, fast computation, and ease of implementation. Recent works further extended random forests to create Hough forests [78–80] for detection, tracking and action recognition of people. A number of related works regarding detection using Hough forest can be found in [81–85]. Furthermore, the tracking method based on Hough forest can be found in [86–88]. and applications to action recognition in a number of studies, [89–91].

A Hough forest is a set of decision trees and each tree provides a mapping from local appearance of an image to the Hough space, $\mathcal{H}$. Local appearance can be denoted by $\mathcal{I}, I^1(y), ...I^F(y)$. The center of the local appearance or local patch is denoted by $\mathbf{y}$. Each of $I^f$ is a feature channel and F is the number of feature cahnnels. The Hough space describes an object of action position in scale, space and class. After training, the leaves of the random forests $\mathcal{L}$ provides a mapping from local patches to the probabilistic Hough votes.

$$\mathcal{L} : (\mathbf{y}, \mathcal{I}) \rightarrow p(\mathbf{h}|L(\mathbf{y})) \tag{3.3}$$

where $p(\mathbf{h}|L(\mathbf{y}))$ is the distribution of the Hough votes within the Hough space $\mathcal{H}$. The detection process is illustrated in Figure 3.3.

Regression forests have been used for body part detection and tracking of humans on the Kinect depth data [14]. This is achieved by employing classification trees which will be summarized below. Human tracking using classification trees is based on a modeling a large number of body parts on depth data using decision trees. Figure 3.4(a) shows the captured depth data of a person from Kinect. Figure 3.4(b) describes the different body regions on the depth data. Figure 3.4(c) demonstrates the body part representations using depth features.

The goal is to determine the class of each depth pixel $\mathbf{p}$ in the depth data captured by Kinect. The classes are the different body parts, denoted by c ∈ { **left hand**, **right hand**, **head**, **left shoulder**, **right shoulder**,... }

Visual features are defined by depth comparisons between pairs of pixel locations. For each reference point $\mathbf{p}$, feature vector $\mathbf{v} = (x_1, ....., x_d)$ is calculated by using the

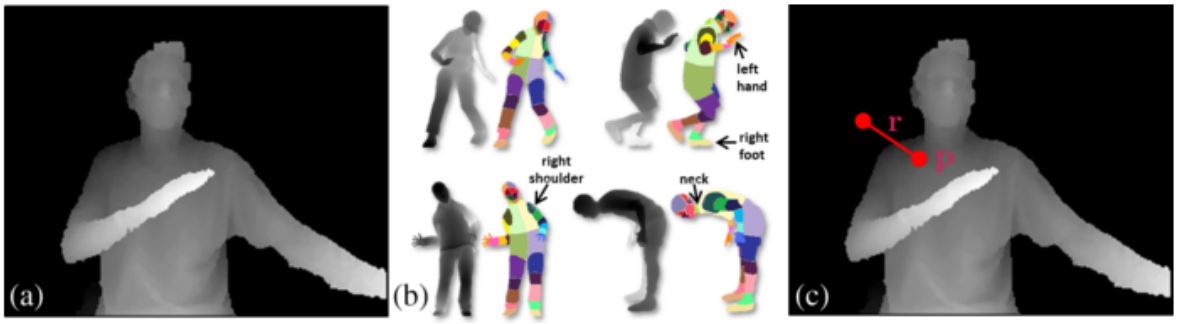



**Figure 3.4:** The body parts detection of a human on a captured Kinect depth data. (a) Acquired Kinect depth data. (b) Ground-truth labelling of 31 different body parts. (c) The description of a feature vector [14].

formulation which is defined in [14] and given by

$$x_i = J(\mathbf{p} - J(\mathbf{p} + \frac{r_i}{J(\mathbf{p})})) \tag{3.4}$$

where the depth value (distance from camera) is denoted by $J(.)$. The 2D position of the depth value is denoted by $\mathbf{p}$. The displacement from reference point $\mathbf{p}$ is denoted by 2D vector $\mathbf{r}$.

**Forest Training** Each tree in the forest is trained by randomly selecting 2000 example pixels. These pixels are passed through a set of trees. At each non leaf node, splitting candidates $\theta = (\mathbf{r}, \tau)$ are selected by maximizing the information gain and the pixels $Q(I, x)$ at each non leaf node are split into left $Q_l$ and right node $Q_r$. The displacement vector $\mathbf{r}$ is randomly selected and the learned threshold is denoted by $\tau$.

Left set can be defined by

$$Q_l(\theta) = \{(\mathbf{I}, \mathbf{x}) f_\theta(\mathbf{I}, \mathbf{x}) < \tau\} \tag{3.5}$$

Right set can be defined by

$$Q_r(\theta) = Q \setminus Q_l(\theta) \tag{3.6}$$

The determination of splitting candidates is based on randomly generating a number of values for these parameters and then selecting the ones that result in maximum

information gain [14].

$$\theta^* = \text{argmax}G(\theta) \tag{3.7}$$

$$G(\theta) = H(Q) - \sum_{s\in\{l,r\}} \frac{|Q_s(\theta)|}{|Q|} H(Q_s(\theta)) \tag{3.8}$$

where

$$H(Q) = \sum_{c\in C} p(c)\text{log}p(c) \tag{3.9}$$

$G(\theta)$ is the information gain. $H(Q)$ is the Shannon entropy and it is calculated using the normalized histogram of body parts.

**Forest Testing** The aim is to classify pixel $\mathbf{x}$ in image $\mathbf{I}$. This can be achieved by passing the depth data through each tree. The tests at each non leaf node is evaluated. When the leaf node is reached, the prediction of the body part class is calculated using stored distribution. Other predictions are also obtained from other trees and the final prediction is calculated, [14], by

$$P(c|\mathbf{I},\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c)|\mathbf{I},\mathbf{x}) \tag{3.10}$$

Recent studies by Fanelli et al. [11,12,18] proposed a method which allowed head pose estimation from 3D scans of the faces using a random regression forest. The proposed method was an adaptation of the work by Criminisi et al [15] for head pose estimation. The work has been proposed for organ detection and localization in 3D computed tomography scans (CT) using random regression forests. Fanelli et al. achieved continues head pose parameter predictions from 3D face scans. Similarly, Criminisi et al. achieved detecting and obtaining bounding boxes of unknown organs within CT scans. In [15], a tree in the regression forest was created by passing a subset of 3D CT scans together with 3D bounding boxes of organs. The tests were defined to guide 3D locations of organs to a left or right leaf node. The tests at each leaf node were defined in equation 3.11. These tests at each leaf node were determined by maximizing the information gain which was based on differential entropy. Similarly, a tree in a regression forest was created by

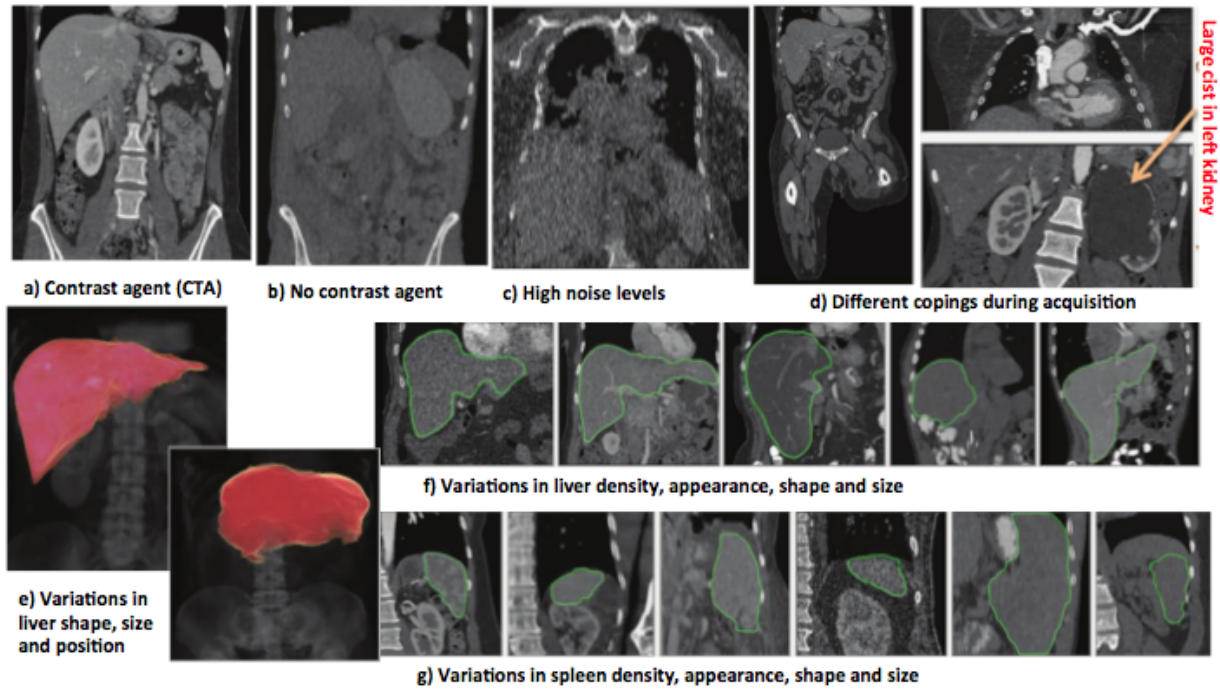



**Figure 3.5:** Organ variability in CT scans. **(a,b,c)** Appearance variation. **(d)** Image geometry variation due to acquisition parameters. **(e)** Different renderings of liver. **(f,g)** Several scans of mid-coronal views of liver and spleen, [15].

passing a subset of 3D face scans together with head pose parameters. The tests were defined to guide fixed sized 3D face patches to a left or a right leaf node. The tests at each leaf node were defied in equation 3.11. The feature responses in 3.11 were defined by mean intensities of over 2D regions within the patches. The test parameters were determined by maximizing the information gain which was based on differential entropy. As a result, the studies have shown that regression forests have provided accurate mapping from 3D data (face/organ) to the continues parameters (3D bounding box or head pose parameters). The more details of a regression forest for detection and localization of anatomic structures on tree-dimensional computed tomography (CT) scans [15] can be given as follows. These structures were heart, liver, spleen, left lung, right lung, left kidney, right kidney, gall bladder, left pelvis and right pelvis. The variations of organs in terms of size, position, shape and appearance can be seen in Figure 3.5. These organs are defined by a tree dimensional bounding box and this bounding box is defined by $\mathbf{b}_c = \{b_c^L, b_c^R, b_c^A, b_c^P, b_c^H, b_c^F\}$ where each value defines the position (in mm) of the corresponding axis-aligned wall. One of the bounding box which defines a left kidney on CT scan can be seen in Figure 3.6.

**Forest Training** The construction of each tree in the forest is performed by passing a subset of organs and corresponding bounding boxes through the tree. The binary tests $\xi > f(\mathbf{v}; \theta_\mathbf{j} > \tau)$ are determined at each non leaf node. The feature response is denoted by f(.) which is computed at a voxel $\mathbf{v} = (v_x, v_y, v_z)$. The voxel represents the 3D location of a CT scan. The tests allow dividing a set into left and right non leaf node. The visual feature at jth non leaf node is denoted by $\theta_\mathbf{j}$. The feature response is defined by

$$f(\mathbf{v}; \theta_j) = |F_1|^{-1} \sum_{q \in F_1} \mathbf{q} - |F_2|^{-1} \sum_{q \in F_2} I(\mathbf{q}) \tag{3.11}$$

The voxels $\mathbf{v}$ and the corresponding offsets $\mathbf{d}_c$ are determined. The offset is defined by $\mathbf{d}_c = d_c^L, d_c^R, d_c^A, d_c^P, d_c^H, d_c^F$. The calculation of offset vectors can be performed by $\mathbf{b}_c = \hat{v} - \mathbf{d}_c\mathbf{v}$ where $\hat{\mathbf{v}} = (v_x, v_x, v_y, v_y, v_z, v_z)$.

The test parameters at non leaf nodes of the trees are calculated by generating random values of these parameters and then selecting the values which provides maximum information gain. The information gain IG is defined by

$$IG = H(S) - \sum_{i=\{L,R\}} w_i H(S_i) \tag{3.12}$$

where H(S) denotes entropy of the set S. The distribution of vectors at each node is modelled by a multivariate Gaussian, $p(d) = N(d_c)$. The differential entropy of this can be defined by

$$H(S) = \frac{n}{2}(1 + \log(2\pi)) + \frac{1}{2}\log|\Lambda(S)| \tag{3.13}$$

and information gain is defined by

$$IG = \log|\Lambda_c S| - \sum_{i=\{L,R\}} w_i \log|\Lambda(S_i)| \tag{3.14}$$

The above information gain is reformulated in order to model six classes of organs. This new information gain is denoted by

$$IG(\phi) = \log|\Gamma(S)| - \sum_{i=\{L,R\}} w_i \log|\Gamma(S)|, \quad \text{with} \quad \Gamma(S) = \text{diag}(\Lambda_1, ..., \Lambda_c, ...\Lambda_{|C|}) \tag{3.15}$$

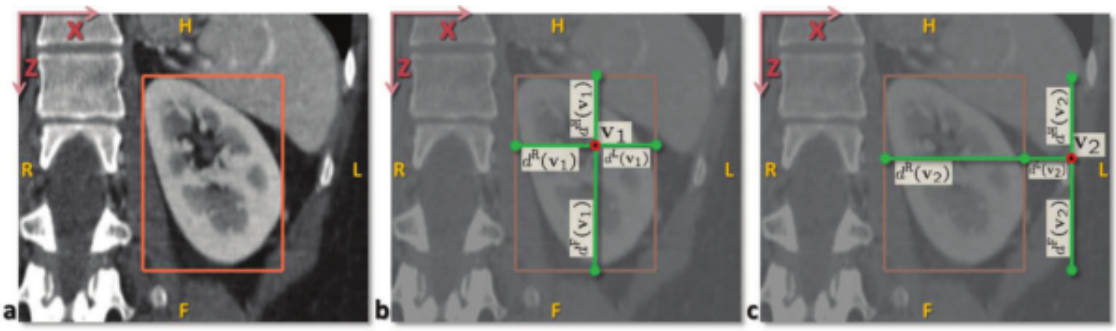



**Figure 3.6:** Representation of a organ in three dimensional CT scans. **(a)** A coronal view of a left kidney and the corresponding bounding box. **(b,c)** The location of voxel $v_i$ and 6 displacemnts from v to boundaries of bounding box, [15].

Maximizing this IG gain allows more accurate casting of probabilistic votes for the positions of a bounding box. After, the parameters are stored at each non leaf nodes. The mean and covariance are also stored at leaf nodes.

**Forest Testing** Anatomy detection and localization is performed as follows. Each voxel $\mathbf{b} \in V$ is sent through each tree. The stored tests at each leaf node guides this voxel towards a leaf node. When the voxel reaches the leaf node, a probabilistic prediction is obtained from the stored distribution $p(\mathbf{d}_c|l) = \mathcal{N}(\mathbf{d}_c; \bar{\mathbf{d}}_c, \mathbf{\Lambda}_c)$. The posterior probability for $\mathbf{b}_c$ is defined by,

$$p(\mathbf{b}_c) = \sum_{l \in \bar{\mathcal{L}}} p(\mathbf{b}_c|l)p(l) \tag{3.16}$$

where $\bar{\mathcal{L}}$ denotes a subset of all forest leaves and $p(l) = 1/|\hat{\mathcal{L}}|$. The organ class is determined if $p(\mathbf{b}_c = \bar{\mathbf{b}}_c) > \beta$ and the position of the organ $\bar{\mathbf{b}}_\mathbf{c}$ which belongs to the $c^{th}$ class is calculated by expectation $\int_{\mathbf{b}_c} \mathbf{b_c} p(\mathbf{b_c}) d\mathbf{b_c}$.

## 3.4 Offline Support Vector Regression

One of the well known batch learning techniques is the Support Vector Regression (SVR), [92, 93]. SVR allows continuous data modeling and closely related to Support Vector Machines. Training is carried out by using a set of training samples. When a new training sample is available, the SVR model is re-trained using all available training samples.

Support vector regressions are further extended to Tensor based approaches [62, 94–97]. Tensor based learning approaches, are known as more powerful than vector-based approaches. The use of vector representations results in several problems. The structure information is lost and the high dimensional data might cause over fitting problems to appear. In [96], supervised tensor learning which models the projection vectors in each mode independently is proposed. However, this approach might not provide adequate discriminate information in some case. As a result, this approach is further extended in studies, [62, 94–96]. In [62], tensor learning for regression is proposed. Two mapping functions are learned using Canonical (CANDECOMP)/Parallel factors (PARAFAC) decomposition, [98]. The square loss and e-sensitive loss functions are studied together with Frobenius norm. In [62], tensor-based regression is compared to Support Vector Regression specifically for the problem of head pose estimation in the IDIAP [99], Boston University dataset, [64] and Pointing4 [65] datasets. The results on these datasets shows that tensor based regression (higher rank Support Tensor Regression ) performed better than other regression algorithms for head pose estimation.

The brief overview of the SVR is given here. Further details can be found in [92]. A linear regression function can be defined by

$$\mathbf{f}(\mathbf{x}) = \mathbf{W^T}\mathbf{\Phi}(\mathbf{x}) + b \tag{3.17}$$

where $\mathbf{x}$ is the input vector, $\mathbf{\Phi}(x)$ is a mapping function maps $\mathbf{x}$ to a vector, $\mathbf{W}$ is the vector weights, b is the bias, and $\mathbf{f}(\mathbf{x})$ is the regression output. The equation is determined by calculating an optimization problem:

$$\begin{aligned} \min_{\mathbf{W},b} P &= \frac{1}{2}\mathbf{W^T}\mathbf{W} + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ s.t. \quad y_i - (\mathbf{W^T}\mathbf{\Phi}(\mathbf{x}) + b) &\leq \varepsilon + \xi_i \\ (\mathbf{W^T}\mathbf{\Phi}(\mathbf{x}) + b) - y_i &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, \quad i &= 1...l. \end{aligned} \tag{3.18}$$

The above optimization can also be calculated using dual optimization by introducing Lagrange multipliers $\alpha, \alpha^*, n$ and $n^*$. Lagrangian is given by

$$\begin{aligned} L_p = \frac{1}{2}\mathbf{W^T W} + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}(n_i\xi_i + n_i^*\xi_i^*) \\ - \sum_{i=1}^{l}\alpha_i(\varepsilon + \xi_i + y_i - \mathbf{W^T}\mathbf{\Phi}(\mathbf{x_i}) - b) \\ - \sum_{i=1}^{l}\alpha_i^*(\varepsilon + \xi_i^* - y_i + \mathbf{W^T}\mathbf{\Phi}(\mathbf{x_i}) + b) \\ s.t. \quad \alpha_i, \alpha_i^*, n_i, n_i^* \geq 0. \end{aligned} \tag{3.19}$$

The dual optimization is given by

$$\begin{aligned} \min_{\alpha,\alpha^*} D = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}Q_{ij}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) + \varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) \\ - \sum_{i=1}^{l}y_i(\alpha_i - \alpha_i^*) \\ s.t \quad 0 \leq \alpha_i, \alpha_i^* \leq C \quad i = 1, .....l, \\ \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \end{aligned} \tag{3.20}$$

where $Q_{ij} = \Phi(x_i)^T\Phi(x_j) = K(x_i, x_j)$. $K(x_i, x_j)$ is a kernel function.

## 3.5 Online Support Vector Regression

The above stated offline learning or batch implementation of the learning algorithms might not be convenient for some applications. The main reason is that for each new batch of training data, the models need to be retrained. Therefore, incremental learning techniques are proposed to address the problems caused by offline methods. The initial online version of SVMs are introduced in [100]. Following [100], [17, 101, 102] studies propose an online versions of the work, [92]. These incremental versions of the methods update the models as the new training samples become available. A successful application of the online support vector regression is reported in [103]. Other online learning based methods have also been developed and are available such as online discriminative kernel density estimation, [104] on-line random forests, [105] and incremental support

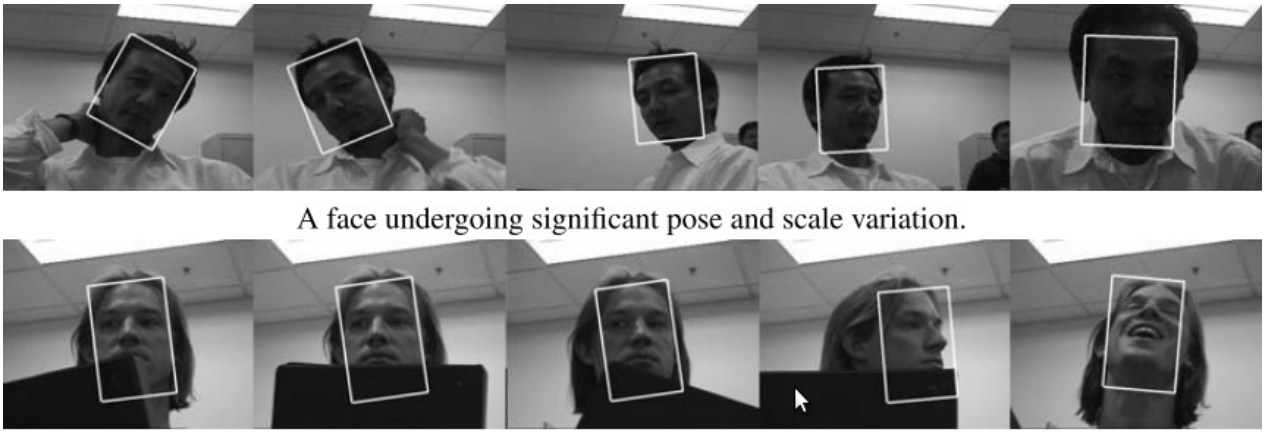



**Figure 3.7:** An example of incremental face tracking on images under occlusion and pose variation, [16].

vector machines [106], [107]. The main advantage of incremental methods like other online methods is that training is performed by adding or removing a sample from the training set without retraining the models from scratch for each new sample. PCA based incremental learning methods are proposed in the previous studies [16, 108, 109]. Figure 3.7 shows tracking results under occlusions and pose variation.

With the SVM framework, the above offline (batch) learning methods can further be extended to online learning methods. More specifically, the regression function can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}) + b \tag{3.21}$$

The Lagrange formulation can also be redefined by

$$\begin{aligned} L_D = \frac{1}{2}\sum_{i=1}^{l}(\alpha_i - \alpha_i^*(\alpha_j - \alpha_j^*) + \varepsilon\sum_{i=1}^{l}(\alpha + \alpha_i^*) - \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \\ - \sum_{i=1}^{l}(\delta_i\alpha_i + \delta_i^*\alpha_i^*) + \sum_{i=1}^{l}[u_i(\alpha_i - C) + u_i^*(\alpha_i^* - C)] \\ +\zeta\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) \end{aligned} \tag{3.22}$$

where $\delta_i^{(*)}, u_i^*$, and Optimizing Lagrange multipliers lead to

$$\begin{aligned} \frac{L_D}{\alpha_i} &= \sum_{j=1} lQ_{ij}(\alpha_j - \alpha_j^*) + \varepsilon - y_i + \zeta + \delta + u_i = 0 \\ \frac{L_D}{\alpha_i^*} &= -\sum_{j=1} lQ_{ij}(\alpha_j - \alpha_j^*) + \varepsilon - y_i - \zeta - \delta + u_i^* \\ \delta_i^*, &\geq 0, \quad \delta_i^* \alpha_i^* = 0 \\ u_i^(*) &\geq 0, \quad u_i^*(\alpha_i^* - C) = 0 \end{aligned} \tag{3.23}$$

KKT conditions allows defining a coefficient which is defined by

$$\theta_i = \alpha_i - \alpha_i^*, \tag{3.24}$$

A margin function can be defined for the ith sample $x_i$ and defined by

$$h((\mathbf{x_i}) = f(x_i) - y_i = \sum_{j=1}^{l} Q_{ij}\theta_j - y_i + b \tag{3.25}$$

Then, the above equations can be used to obtain

$$\begin{aligned} h(\mathbf{x_i}) &\geq \varepsilon, \quad \theta_i = -C \\ h(\mathbf{x_i}) &= \varepsilon, \quad -C < \theta_i < 0 \\ -\varepsilon \leq h(\mathbf{x_i}) &\leq \varepsilon, \quad \theta_i = 0 \\ h(\mathbf{x_i}) &\geq \varepsilon, \quad 0 < \theta_i = C \\ h(\mathbf{x_i}) &\leq -\varepsilon, \quad \theta_i = C \end{aligned} \tag{3.26}$$

The above five conditions can be used to define three subsets which are support, error and remaining set. The partitioning of the these sets is performed by satisfying the Karush Kuhn Tucker (KKT) condition. The KKT condition can be defined by $\theta_i, h_i(x_i), \varepsilon, C$. $\theta$ is defined in terms of dual variables. C is used to control the penalization on the norm of weights. The margin function is defined by $h_i(x_i)$ as the difference $f(x_i) - y$ for all training samples. Training data is partitioned into three sets results in

support, remaining and error sets. These sets are defined by

$$\text{Error support vectors:} \quad E = \{i| \quad |\theta_i| = C\} \tag{3.27}$$

$$\text{Margin support vectors:} \quad S = \{i| \quad 0 < |\theta| < C\} \tag{3.28}$$

$$\text{Remaining samples:} \quad R = \{i| \quad \theta_i = 0\} \tag{3.29}$$

Incremental update of the SVR function is carried out when a new sample is available. The values of $\theta$ are changed incrementally until the KKT conditions are satisfied.

A decremental (unlearning) algorithm is also possible with this online SVR. If the training sample removed from the training set, this is called decremental step. The decremental process is only carried out for training samples which are in the remaining set. This is performed by setting the value of coefficient $\mathbf{x}$ to zero then adjusting the KKT conditions for all other samples. Figure 3.8 shows the support, remaining and error sets. The yellow circles are support samples, the green circles are remaining samples and the red circles are error samples.

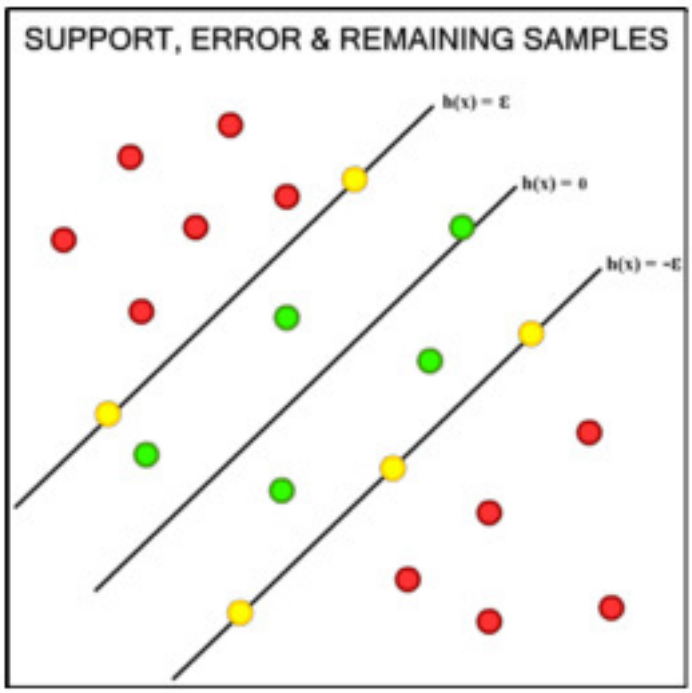



**Figure 3.8:** The online Support Vector Regression model. The support, remaining and error sets. The yellow circles are support samples, the green circles are remaining samples and the red circles are error samples, [17].

## 3.6 Principal Component Imagery

In the proposed approach, eye images were represented in eignspace. Representing the eye images in eignspace allowed defining eye images in low dimensional vectors. Low dimensional vector representations have two advantages in the proposed method. First, modeling the eye images using a Gaussian Model (GMM) required less computation time. Second, updating online Support Vector Regression (OSVR) models online also resulted in less computation time. As a result, the unified system which made use of a GMM and OSVR models allowed online learning of gaze parameters in real-time.

Let $I_1, I_2, .....I_L$ be a set of images. Given this set of images, a set of vectors, $\mathbf{x}_1, ...\mathbf{x}_L$ can be obtained by reordering pixel values of these images, [110]. PCA allows modelling the distribution of a set of vectors using a set of L orthonormal vectors, $\Phi_L$, and their corresponding eigenvalues, $\Lambda_L$. The calculation of these set of L orthonormal vectors is as follows. The mean and covariance matrix of the vectors are calculated. Then, the eigenvalue problem is solved. The eigenvalue problem is defined by

$$\Lambda = \Phi^T \Sigma \Phi \tag{3.30}$$

where $\Sigma$ is the covariance matrix of the data, $\Phi$ is the eigenvectors, and $\Lambda$ is the eigenvalues. N vectors with high eigenvalues are selected. Then, principal component feature vectors are calculated by $\mathbf{y} = \Phi_N^T \hat{\mathbf{x}}$ where $\hat{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ mean normalized image vector. The mean vector of the training data denoted by $\bar{\mathbf{x}}$.

## 3.7 Gaussian Mixture Models

The Gaussian mixture model is a weighed sum of M component Gaussian densities and defined by

$$p(\mathbf{y}|\Theta) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{y}|\theta_i) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{y}|\mu_i, \Sigma_i) \tag{3.31}$$

where $\theta_i = (\mu_i, \Sigma_i)$ is the set of parameters of the i-th mixture component, and $\alpha_i$ is the weight of the mixture component. $\Theta = \{\alpha_i ..., \alpha_M, \theta_1, ....\theta_M\}$ is the set of all parameters in the mixture. The number of mixture components is denoted by M. In Gaussian

mixture model, each Gaussian density is denoted by

$$\begin{aligned} p(\mathbf{y}|\theta_i) &= p(\mathbf{y}|\mu_\mathbf{i}, \mathbf{\Sigma_i}) \\ &= \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{y}-\mu)\Sigma_i^{-1}(\mathbf{y}-\mu_\mathbf{i})\} \end{aligned} \tag{3.32}$$

Assuming that the given projection vectors $\mathbf{Y} = \{\mathbf{y}_i, ....\mathbf{y}_N\}$ are independent and identically distributed, the joint density of the vectors can be defined by

$$p(\mathbf{Y}|\Theta) = \prod_{j=0}^{M} \alpha_i \mathbf{p_i}(\mathbf{y}|\theta_\mathbf{i}) = \mathcal{L}(\Theta|\mathbf{Y}) \tag{3.33}$$

where $\mathcal{L}(\Theta)$ is the likelihood function. Similarly, $\log\mathcal{L}(\Theta|X)$ is called log likelihood function and defined by

$$log(\mathcal{L}(\Theta|Y)) = \sum_{i=0}^{N} log\bigg(\sum_{j=0}^{M} \alpha_i \mathbf{p_i}(\mathbf{x}|\theta_\mathbf{i})\bigg) \tag{3.34}$$

The aim is to find the Gaussian mixture parameters, $\mathbf{\Theta} = \{\alpha_\mathbf{i}..., \alpha_\mathbf{M}, \theta_\mathbf{1}, ....\theta_\mathbf{M}\}$, which models the joint density of the samples. This is achieved by the maximizing log-likelihood expression of the training set. The parameter calculation, maximizing log likelihood function, is achieved using Expectation-Maximization algorithm, [111, 112]. The EM algorithm calculates parameters in two steps which are E-step and M-step. In E-step, the expected value of complete data is calculated and defined by

$$Q(\Theta, \Theta^{i-1}) = E \log p(\mathcal{X}, \mathcal{Y})|\mathcal{X}, \Theta^{(i-1)}) \tag{3.35}$$

In the M-step, the Gaussian mixture model's parameters are optimized

$$\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{i-1}) \tag{3.36}$$

The most probable mixture component of each projection vector is calculated using a posteriori probability of each mixture component and then selecting the one which has maximum probability. A posteriori probability of one mixture component is defined by

$$p(i|\mathbf{y}_t, \alpha) = \frac{w_i p(y_t|\mu, \Sigma))}{\sum_{k=1}^{M} w_k p(y_t|\mu_k, \Sigma_k)} \tag{3.37}$$

## 3.8 Conclusions

The proposed online gaze estimation method for updating models for more accurate gaze estimates was introduced. The proposed method was based on modeling eye images by a Gaussian Mixture Model for test eye patch modeling. The new test eye patches were also used to update Support Vector Regressions. The advantages of online Support Vector Regression over Support Vector regression was described in detail. The main advantage of online Support Vector Regression was that it allowed for fast model update and result in real time gaze estimates. Finally, representing eye images in lower dimensional data resulted in fast data modeling and model updates using OSVR. All methods which were used in a unified method was also summarized in a Table. In chapter 5, the proposed methods based on online gaze estimation were introduced.

The proposed head pose estimation method was also combined with the proposed gaze estimation system. The proposed random forests and tensor regression based system was also compared with classic random forests. The advantages of the method over random forests were reported. The proposed head pose estimation method allowed eliminating the distortion on eye images due to head pose variation. The new head pose estimation method which was based on random forests and Tensor regressors at each leaf node will be described in the next chapter. Therefore, an overview of random forests and their related work was provided in this chapter. Furthermore, Support Vector Regressions (SVR) and Tensor learning for regression were described. As a result, this chapter provided an overview of the methods which were used in the proposed gaze estimation and head pose estimation.

# Chapter 4

# Real-time Head Pose Estimation

## 4.1 Introduction

In this chapter, we introduce a novel method called random forest based tensor regression, for real-time head pose estimation using both depth and intensity data. The method builds on random forests and proposes to train and use tensor regressors at each leaf node of the trees of the forest. The tensor regressors are trained using both intensity and depth data and their votes are fused. The proposed method is shown to outperform current state of the art approaches in terms of accuracy when applied to the publicly available Biwi Kinect head pose dataset.

We address the problem of head pose estimation by extending previous methods of random forest based head pose estimation in three ways: i) by using tensor-based regression at each leaf node, ii) by fusing depth and intensity data using tensor regression at each leaf node, and iii) by fusing depth and intensity data using a random forest framework. This approach differs to classical random forests in which the prediction models at each leaf node disregard the data of the sample that arrives at the leaf in question. Indeed, typical random forest methods employ leaf node prediction models that rely on the statistics of the training samples that arrive at the leaf in question. Typically the mean, and sometimes also the covariance matrix are used. Here, we propose to use stronger regression models in order to estimate parameters more accurately. We specifically propose to use Tensor regression models as they have shown good generalisation properties when the availability of data is sparse. This is often the case in our work, where the number of the samples that arrive at several leaf nodes can be small. In Figure 4.1, we give an outline of the proposed method. First, a regression forest is

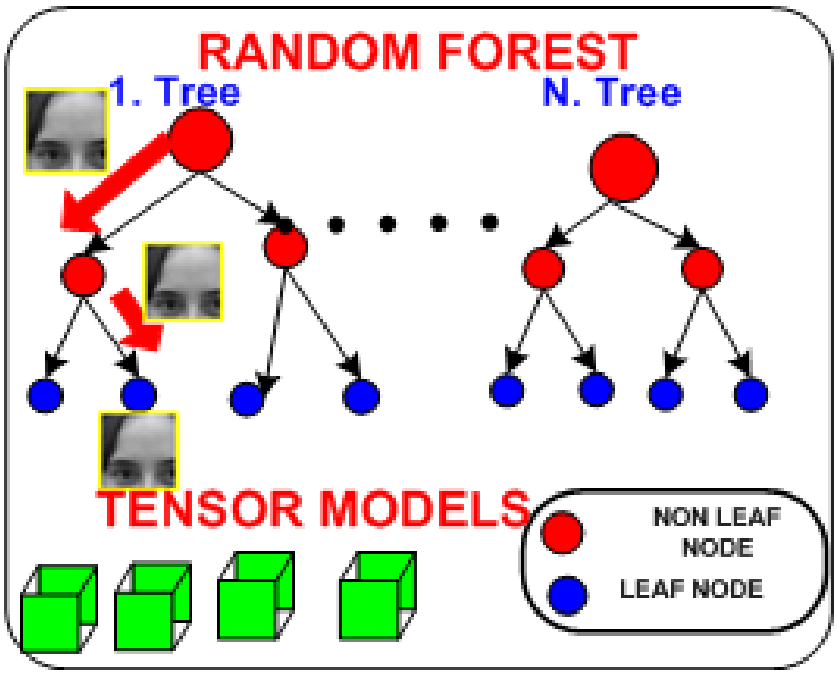



**Figure 4.1:** Multimodal Random Forest based Tensor Regression. Fixed size patches are extracted from depth and intensity patches and passed through the forest. The patches which reach the leaf nodes are used as inputs to the tensor based regression models to obtain the estimates of the head angles.

learned by passing randomly extracted depth patches with head pose parameters into the tree. At each non-leaf node each patch is sent either to the left or to the right branch according to the result of a test on features extracted from the patch in question. The test is chosen at training time as the one that maximises the information gain on the head pose parameters that will be achieved if the test in question is applied on the training patches that arrive at the internal node in question. Once a test is chosen, it is applied to the training patches that arrive at that internal node. This is repeated recursively until a stopping criterion is met. In this Chapter, we propose to learn a tensor regression model at each leaf node, and report results on three variations of tensor regression models. The first one, termed RF-TR-D, uses patches extracted from depth images. The second one, termed RF-TR-I, uses patches extracted from grayscale images, and the third one, termed RF-TR-ID, fuses the information from RF-TR-D and RF-TR-I. In addition, integration of intensity and depth data in a random forest framework is exploited. Integration allows usage of both intensity and depth information for more accurate parameter estimation. The robustness and accuracy of this method is proven over random forest based regression technique employing only depth data for the Real-time large head pose estimation experiments.

In summary, this Chapter contributes two novel approaches to the problem of real-time head pose estimation. First, a new regression method based on random forests and tensor models is proposed. The proposed method instead of modelling the votes at each leaf node with a Gaussian, employs a tensor-based regression model. Second, we proposed fusion of depth and intensity data in the random forest regression framework.

The remainder of this Chapter is as follows. In Section 4.2, we introduce the Multimodal Random Forest based Tensor Regression method. In Section 4.3, we show how the proposed method is applied to the problem of head pose estimation. Results are given in Section 4.4 and conclusions are drawn in Section 4.5.

(a) (b)

**Figure 4.2:** Aligned depth data and gray scale images. (a) Captured depth data. The green bounding box shows head/face region in which fixed size depth patches are extracted. The yellow bounding box shows the extracted fixed size depth patch. (b) Corresponding gray scale image. The green bounding box shows head/face region in which fixed size gray scale patches are extracted. The yellow bounding box shows the extracted fixed size gray scale patch.

## 4.2 Multimodal Random Forest Based Tensor Regression

First, a random forest based tensor regression employing only depth data (RF-TR-D) is described. Second, integration of intensity and depth at each leaf node of a random forest using tensor models (RF-TR-ID) is introduced.

A number of aligned grey scale images and depth data with head location and orientation parameters are used for the construction of a forest. An example of the aligned grayscale images and depth data is given in Figure 4.2. For the construction of each tree, a subset of aligned intensity and depth data is selected and several fixed size patches belonging to head/face region are extracted from both the intensity image and the depth data. Fixed size patches (red) are also extracted from the other image areas depicting the torso, the arms and the hair.

We assume that images are annotated in terms of the face bounding box. Given that, we proceed in building a classification and regression forest. The classification of the patches in terms of whether they belong to the background or the face region is performed at the leaf nods of the forest. The ratio of positive (i.e. belonging to the face area) and negative (i.e. not belonging to the face area) patches that arrive at each leaf node during testing is calculated and stored. During testing, when a patch reaches the leaf note, it is classified as positive or negative, depending on whether this ratio is greater than one or not.

### 4.2.1 Tree Construction

Let us denote a random forest by $\mathbf{T}=\{T_t\}$, where a tree in the forest is denoted by $T_t$. Each tree, $T_t$, is built using a set of patches, $\{P_i\}$ , which are randomly chosen from the training data. A patch is denoted by $P_i = (I_i^f, c_i, \theta_i)$ where $I_i^f$ are the extracted features, $c_i$ is a class label that reveals whether the patch belongs to the face/head region ($c_i = 1$) or not ($c_i = 0$) and $\theta = \{\theta_x, \theta_y, \theta_z, \theta_{yaw}, \theta_{pitch}, \theta_{roll}\}$ is a vector which contains the head pose parameters. The values $\theta_x, \theta_y, \theta_z$ are offsets between the patch center and the head center in 3D and $\theta_{yaw}, \theta_{pitch}, \theta_{roll}$ are the Euler angles of the head pose parameters.

The tree is constructed using the method proposed in [12], [18]. The parameters of a tree in each internal node are selected by generating a number of binary tests and by selecting the best test according to the optimization criterion. In this work, we use stumps tests with parameters $t_{F_1,F_2,\tau}$. That is:

$$\frac{1}{\mid F_1 \mid} \sum_{q \epsilon F_1} I^f(q) - \frac{1}{\mid F_2 \mid} \sum_{q \epsilon F_2} I^f(q) > \tau \tag{4.1}$$

where $I^f$ is a feature channel and $F_1$ and $F_2$ are rectangular regions determined randomly within the depth patch. $\tau$ denotes the threshold. Similarly to [12, 15], our tests use the

differences between the average values of rectangular regions. The optimization function is defined using both classification and regression measure. The function is optimized by randomly choosing between a measure based on the classification performance (face / not face) and a measure based on the regression performance (i.e. related to the estimation of the head pose) at each non-leaf node.

More specifically, the classification measure is defined as

$$U_C(P \setminus t^k) = \sum_{i \in \{L,R\}} w_i H(P \setminus t^k) \tag{4.2}$$

where

$$H(P \setminus t^k) = p(c|P) ln(p(c|P)) \tag{4.3}$$

The set of patches at the parent node i is denoted by $P_{i\epsilon\{L,R\}}$ and the sets of patches at the child nodes are denoted by $P_L$ and $P_R$. The ratio of patches is denoted by $w_i = \frac{|P_i|}{|P|}$.

The regression measure is defined as the differential entropy of the set of patches $P$ at the internal node minus the weighted sum of differential entropies at the left child node $P_L$ and the right child node $P_R$, which are defined after the splitting process. That is:

$$U_R(P \setminus t^k) = H(P) - (w_L H(P_L) + w_R H(P_R)). \tag{4.4}$$

where H(P) is the differential entropy of the set $P_{i\in\{L,R\}}$ and $w_{i=L,R}$ is the ratio of patches sent to each child node. The equation then becomes:

$$U_R(P \setminus t^k) = log(\Sigma^v + \Sigma^a) - \sum_{i=\{L,R\}} w_i log(\Sigma_i^v + \Sigma_i^a) \tag{4.5}$$

where $\Sigma^v$ and $\Sigma^a$ are the covariance matrices of the offset vectors and rotation angles.

The splitting process stops and a leaf node is declared when the number of patches is below a threshold or when the tree reaches a predefined depth level.

In this study, fixed sized patches (60x60) are extracted from the input data. Scale variation is not considered since we assume that the subjects are approximately at the same distance to the camera. This is a valid assumption in the Biwi Head pose

**Figure 4.3:** RGB images and depth data from the Biwi Kinect Head Pose Database, [18].

dataset, [18] which we use for performance evaluation. Some example images from this dataset can be seen in Figure 4.3.

### 4.2.2 Tensor Regression at the Leaf Nodes

Typically, in random regression forests, a multivariate Gaussian distribution models the distribution of the head pose parameters that corresponds to the patches that arrive at the leaf in question. The parameters of each Gaussian are then stored at each leaf node.

In this study, we propose to use tensor regression modes [62] instead of a multivariate Gaussian distribution for regression. Tensor regression is the extension of Support Vector Regression (SVR), [92, 93]. The SVR is performed in the vector space while tensor regression is performed in the tensor space. A tensor can be defined by a multidimensional or N-way array. For example, gray scale image is a 2-way array while rgb image is a 3-way tensor.

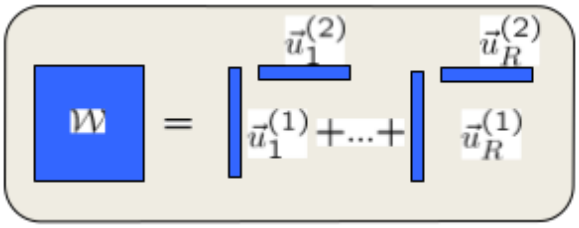
The linear regressor in the vector scape can be defied by

$$y = f(\mathbf{x}; \mathbf{w}, b) = \langle \mathbf{x}, \mathbf{w} \rangle + b, \tag{4.6}$$

where $\mathbf{x}$ is the input data in a vector formant, $\mathbf{w}$ is the weight vector, b is the bias, and y is the estimated scalar output value.

In the tensor space,

$$y = f(\mathcal{X}; \mathcal{W}, b) = \langle \mathcal{X}, \mathcal{W} \rangle + b, \tag{4.7}$$




(a)

**Figure 4.4:** The CANDECOMP/PARAFAC decomposition of a two-way array.

where in our case $\mathcal{X} \in \Re^{M \times N}$ denotes the feature channel patch tensor and $\mathcal{W}$ is the weight tensor. M and N are the dimensions of the patches. The scalar $b$ denotes the bias. The tensor learning for regression algorithm can be seen seen in Algorithm 1, [62].

The tensor learning for regression is based on CANDECOMP/PARAFAC decompostion, [62, 98]. The decomposition allows weight tensor to be represented by the sum of R rank-one tensors, that is:

$$\mathcal{W} = \sum_{r=1}^{R} \vec{u}_r^{(1)} \circ \vec{u}_r^{(2)} \circ \cdots \circ \vec{u}_r^{(M)} \triangleq [\![\vec{U}^{(1)}, \vec{U}^{(2)}, \cdots, \vec{U}^{(M)}]\!], \tag{4.8}$$

where $\vec{U}^{(j)} = [\vec{u}_1^{(j)}, \cdots, \vec{u}_R^{(j)}]$.. The CANDECOMP/PARAFAC decomposition of a two-way array can be seen in Figure 4.4.

The equation 4.8 can be substituted in equation 4.7 and the new equation can be defined by

$$\begin{aligned} y &= \langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}} \rangle + b \\ &= \langle \boldsymbol{\mathcal{X}}, \sum_{r=1}^{R} \mathbf{u}_r^{(1)} \circ ... \circ \mathbf{u}_r^{(M)} \rangle + b \\ &= \sum_{r=1}^{R} \langle \boldsymbol{\mathcal{X}}, \mathbf{u}_r^{(1)} \circ ... \circ \mathbf{u}_r^{(M)} \rangle + b \\ &= \sum_{r=1}^{R} \mathcal{X} \prod_{k=1}^{M} \times_k \boldsymbol{u}_r^k + b \end{aligned} \tag{4.9}$$

In equation 4.9, the input 2D patch is projected onto R directions for each mode k. The projection of input 2-way tensor onto R directions allows feature selection or dimensionality reduction before regression process. After, the model parameters ($\Theta$ =

**Algorithm 1 TENSOR LEARNING FOR REGRESSION, [62]**

**Input**: The set of training tensors and their corresponding targets, i.e. $\{\mathcal{X}_i, y_i\}|_{i=1}^N$

**Output**: Weights $\{\mathbf{U}^1, ....., \mathbf{U}^M\}$ and the bias term b $\in \mathcal{IR}$ that minimize the objective function

1: Randomly initialize $\{\mathbf{U}^1, ....., \mathbf{U}^M\}^{(0)}$

2: **repeat**

**for** k=1 to M **do**

3: t $\leftarrow$ t+1

4: Solve with respect to $\mathbf{U}^k|_{(t)}$ the equation 4.13

**end for**

5: **until** $\|\mathbf{W}^{(t)} - \mathbf{W}^{(t-1)}\|/\|\mathbf{W}^{(t-1)}\| \leqslant \varepsilon$ or $t \geqslant T_{max}$

$\{\vec{U}^{(1)}, \vec{U}^{(2)}, \cdots, (\vec{U})^M, b\}$) at each leaf node are learned by minimizing the regularized empirical risk function. This function is minimized by using a set of labelled 2-mode feature channel patch tensors $\{\mathcal{X}_i, y_i\}_{i=1}^N$ and the associated pose angles, $y_i$. The risk function is given by:

$$L(\Theta) = \frac{1}{2}\sum_{i=1}^{N} l(y_i, f(\mathcal{X}_i; \Theta)) + \frac{\lambda}{2}\psi(\Theta) \tag{4.10}$$

where $l(\cdot)$ is the $\epsilon$-insensitive loss function and $\psi(\cdot)$ is the Frobenius norm regularization term. We select the rank $R$ of the tensor by performing cross validation on the data at each leaf node.

At each leaf node of a forest two tensor regression models are trained. One using depth information, and the second using intensity information.

The optimization methodology of SVR is modified for the tenser learning and it can be defined by

$$\begin{aligned} \min_{\mathbf{W},b,\xi,\hat{\xi}} \quad & \frac{1}{2}\|\boldsymbol{\mathcal{W}}\|_{Fro}^2 + C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) \\ s.t. \quad & -y_i + \langle \boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{W}} \rangle + b \geq \epsilon + \hat{\xi}_i \\ & y_i - \langle \boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{W}} \rangle - b \leq \epsilon + \xi_i \\ \epsilon \geq 0, \;\; & \xi_i \geq 0, \;\; \hat{\xi}_i \geq 0, \;\; \forall i = 1, 2, ..., N. \end{aligned} \tag{4.11}$$

The above optimization function can be solved by keeping all $\boldsymbol{U}^k$ values fix and then solving $\boldsymbol{U}^j$, k $\neq$ j

$$\begin{aligned}\min_{\boldsymbol{U}^j,b,\xi,\hat{\xi}} \quad & \frac{1}{2}Tr(\boldsymbol{U}^{(j)}\boldsymbol{U}^{(-j)T}\boldsymbol{U}^{(-j)}\boldsymbol{U}^{(j)T}) + C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) \\ s.t. \quad & -y_i + Tr(\boldsymbol{U}^{(j)}\boldsymbol{U}^{(-j)T}\boldsymbol{X}_{i(j)}^T) + b \geq \epsilon + \hat{\xi}_i, \\ & y_i - Tr(\boldsymbol{U}^{(j)}\boldsymbol{U}^{(-j)T}\boldsymbol{X}_{i(j)}^T) - b \leq \epsilon + \xi_i, \\ & \epsilon \geq 0, \;\; \xi_i \geq 0, \;\; \hat{\xi} \geq 0, \;\; \forall i = 1, 2, ..., N.\end{aligned} \tag{4.12}$$

The above optimization function can be rewritten by denoting $\boldsymbol{B} = \boldsymbol{U}^{(-j)T}\boldsymbol{U}^{(-j)}$, $\tilde{\boldsymbol{U}}^j = \boldsymbol{U}^j\boldsymbol{B}^{\frac{1}{2}}$, and $\tilde{\boldsymbol{X}}_{i(j)} = \boldsymbol{X}_{i(j)}\boldsymbol{U}^{(-j)}\boldsymbol{B}^{\frac{1}{2}}$. The new optimization function can be defined by

$$\begin{aligned}\min_{\boldsymbol{U}^{(j)},b,\xi,\hat{\xi}} \quad & \frac{1}{2}Tr(\tilde{\boldsymbol{U}}^{(j)}\tilde{\boldsymbol{U}}^{(j)T}) + C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) \\ s.t. \quad & -y_i + Tr(\tilde{\boldsymbol{U}}^{(j)}\tilde{\boldsymbol{X}}_{i(j)}^T) + b \geq \epsilon + \tilde{\xi}_i \\ & y_i - Tr(\tilde{\boldsymbol{U}}^{(j)}\tilde{\boldsymbol{X}}_{i(j)}^T) - b \leq \epsilon + \tilde{\xi}_i \\ & \epsilon \geq 0, \;\; \xi_i \geq 0, \;\; \hat{\xi} \geq 0, \;\; \forall i = 1, 2, ..., N.\end{aligned} \tag{4.13}$$

The parameters can be vectorize $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{X}_{i(j)}}$ using the following relations

$$\begin{aligned}&Tr(\tilde{\boldsymbol{U}}^{(j)}\tilde{\boldsymbol{U}}^{(j)T}) = \|\text{vec}(\tilde{\boldsymbol{U}}^{(j)})\|^2 \\ &Tr(\tilde{\boldsymbol{U}}^{(j)}\tilde{\boldsymbol{X}}_{i(j)}^T) = [\text{vec}(\tilde{\boldsymbol{U}}^{(j)})]^T[\text{vec}(\tilde{\boldsymbol{X}}_{i(j)})]\end{aligned} \tag{4.14}$$

After, the optimization function in equation can be solved using SVR optimizer. First, $\boldsymbol{U}$ is solved and then $\boldsymbol{U}$ is calculated by

$$\mathbf{U}^j = \tilde{\mathbf{U}}^j\mathbf{B}^{-\frac{1}{2}}$$

### 4.2.3 Integrating Intensity and Depth Cues using Random Forests

A unified random forest that allows integration of intensity and depth data is generated. Instead of using only depth values as a feature channel, a number of feature channels with head location and orientation parameters are used for the construction of a forest. The construction of a forest is given in Section 4.2.1. These feature channels are obtained

by extracting features from intensity images and combining them with the depth data. Feature channels are generated after aligning intensity and depth data. An example of aligned depth data and gray scale image can be seen in Figure 4.2. Extracted feature channels from intensity images contain raw gray scale values and nine HoG like feature channels. Gray values are used as a feature channel as extra computation to obtain them is not required. HoG like features are used in the fusion process as they provide good estimates for head pose parameters [22]. Depth values are used as one of the feature channels in the fusion process as their values provide accurate estimates with random regression forests [18].

Head pose estimation from integrated intensity images and depth data was performed as described in Section 4.2.2. Instead of densely extracting fixed size patches from depth data, fixed size patches are extracted from feature channels and passed through the forest. The patches which reach the leaf nodes are used as inputs to the Gaussian models to obtain the estimates of Euler angles.

## 4.3 Head Pose Estimation

During testing, fixed size patches are densely extracted from the depth data and passed through the forest. Each extracted patch is directed via binary tests performed at the internal nodes towards leaf nodes.

When a test patch reaches a leaf node, the patch is classified according to whether it comes from the area depicting the head or other body parts. The classification is made by examining the ratio of head to non-head patches that reach the leaf in question during training ($p(c = 1|P)$). At leaves for which the ratio is greater than one, the trained regressor is applied, an estimate for the head pose parameters is obtained and a vote is cast in the corresponding Hough space. Leaf nodes with high variance ($\Sigma^v > 800$) do not cast votes. Then, the mean shift algorithm is applied in order to remove outliers.

For head center estimation, an approach similar to that proposed in [18] is followed. At each leaf node a vote is cast at the mean of a Gaussian estimated at training time. First, the votes are grouped together. Then, a mean shift with a sphere radius equal to the average face model of [113] is applied in order to remove outliers. Then the votes are averaged to obtain the final head center estimate.

As stated above, the fix-sized patches are densely extracted from the depth data and passed to a forest. Each patch reaches a leaf node of a forest and an estimate of head pose parameters is obtained. However, each patch provides different estimates of head pose parameters. A majority of estimates are expected to be the estimates of actual values of head pose parameters. A small number of estimates might not be accurate estimates of the actual values of head pose parameters. The inaccurate estimates might be obtained from the patches which were extracted from the occluded parts of the face. These inaccurate estimates can be avoided. The actual values of head pose parameters are assumed to be the mean of all estimates obtained from all patches. Then the radius can be defined on the center of the estimated mean head pose parameters. The estimates which are within this radius are average to obtained the estimate of head pose parameters. The patches which are outside of this radius do no considered in the averaging process. The patches which provide head pose estimates outside this radius might be occluded patches and they are called outliers. Defining radius around the mean head pose parameters and removing outliers is the mean shift algorithm. Other methods can also be used to remove outliers instead of the mean shift algorithm. In this work, the mean sift algorithm is used because it is easy to implement and it provides fast computation which is crucial for real-time parameter estimation.

## 4.4 Experimental Results

We evaluate the proposed method by conducting experiments on the publicly available Biwi Kinect head pose database [18] and the ICT-3DHP [114]. We report the performance of the variants of the random forest tensor regression, that is, a) using only depth data (RF-TR-D) and b) by fusing depth and intensity data (RF-TR-ID). For comparison we provide results obtained with a baseline Random Forest.

### 4.4.1 Biwi Kinect Database

Biwi kinect database was created using a Kinect sensor. It consists of depth data and RGB images of upper body region of 20 different people (14 men and 6 women) that turn their heads in different directions. 24 sequences were generated while some people were recorded twice. All images are annotated with head center locations and rotation angles. The rotation angles range is approximately between $\pm 75$ ° for yaw, $\pm 60$ ° for

pitch and ±50 ° for roll. The approximate rotation and translation errors of this dataset is reported about 1 mm and 1 degree.

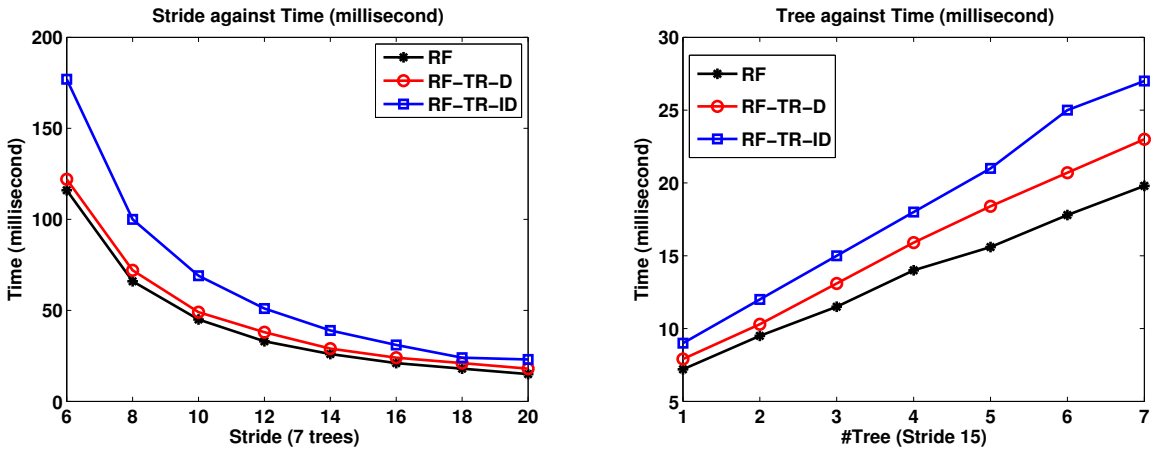
The dataset was partitioned into 18 sequences of 18 subjects as a training set and 2 sequences of 2 subjects as a test set. Two methods were trained using the training set. A random forest was constructed by generating 7 trees. Each tree was generated using 3000 sample images.

### 4.4.2 ICT-3DHP dataset

The ICT-3DHP dataset also contains both depth data and RGB images of the upper body region of different people that turn their heads in different directions. The data were captured by a Kinect sensor. 10 sequences were generated, each containing approximately 1400 frames. All images are annotated with the location of the head center and with the head rotation angles using Polhemus FASTRAK folck of birs tracker.

### 4.4.3 Parameter Setting

For our evaluation, we partitioned the Biwi dataset and used 18 sequences of 18 subjects as training set and 2 sequences of 2 subjects as test set. Each forest contained 7 trees. Each tree was generated using 3000 sample images. The values of the parameters used to train the random forest were set as follows: the sizes of the patches were set to 60x60, and the maximum size of sub patches to 30x30 pixels; the maximum tree depth was set to 15; the minimum number of patches that arrived at an internal node during training was set to 20 and the number of tests at each internal node was set to 10000. The stride was set equal to 5 and the maximum variance to 500. The tensor model's training is performed for each leaf node after the random forest is constructed. Their performance depends on the rank (R) and the regularization parameters (C) used. Their values were selected by cross validation. Figure 4.5 shows head pose parameter computation cost for an average frame in millisecond for the proposed methods, RF-TR-D and RF-TR-ID. Head pose parameter computation time of a given depth data in milliseconds for a random forest with Gaussian models at each leaf nodes are also provided. Figure 4.5(a) depicts the average run time as a function of the stride value. Figure 4.5(b) depicts the average run time as a function of the number of trees when the stride value is 15. As can be seen in Figure 4.5, the proposed method RF-TR-D processes 26 frames per second or higher when the stride value is set to 12 or higher. Similarly, the proposed method RF-




(a) (b)

**Figure 4.5:** Head pose parameter computation time of a given depth and intensity data in milliseconds for the proposed methods, RF-TR-D and RF-TR-ID. Head pose parameter computation time of a given depth data in milliseconds for a random forest with Gaussian models at each leaf nodes are also provided. a) Computation time using 7 trees as a function of the stride parameter. b) Computation time as a function of the number of the trees when the stride value is set to 15.

TR-ID also processes 19 frames per second or higher when the stride value is set to 12 or higher. Although the proposed intensity and depth fusion at each leaf node (RF-TR-ID) results in a slightly higher computation cost than the proposed RF-TR-D method, both methods work in real-time. For the evaluation of the proposed methods on the ICT-3DHP dataset, we used the forest which was generated using the Biwi dataset. The parameter setting which was used during the forest training is described in Section4.4.3.

### 4.4.4 Random Forest with Tensor Models

In this section, we compare the performance of the proposed method with plain random forests. The results can be seen in Figure 4.6. Figure 4.6(a) and (b) reports the percentage of correctly estimated head poses on depth images for different thresholds for head center localization and angular error. In Figure 4.6(c), we report the Mean Angle error (MAE) against the percentage of leaves.

As can be seen in the Figure 4.6(b), the proposed intensity and depth fusion at each leaf node perform similarly for different thresholds. The proposed method, RF-TR-D is

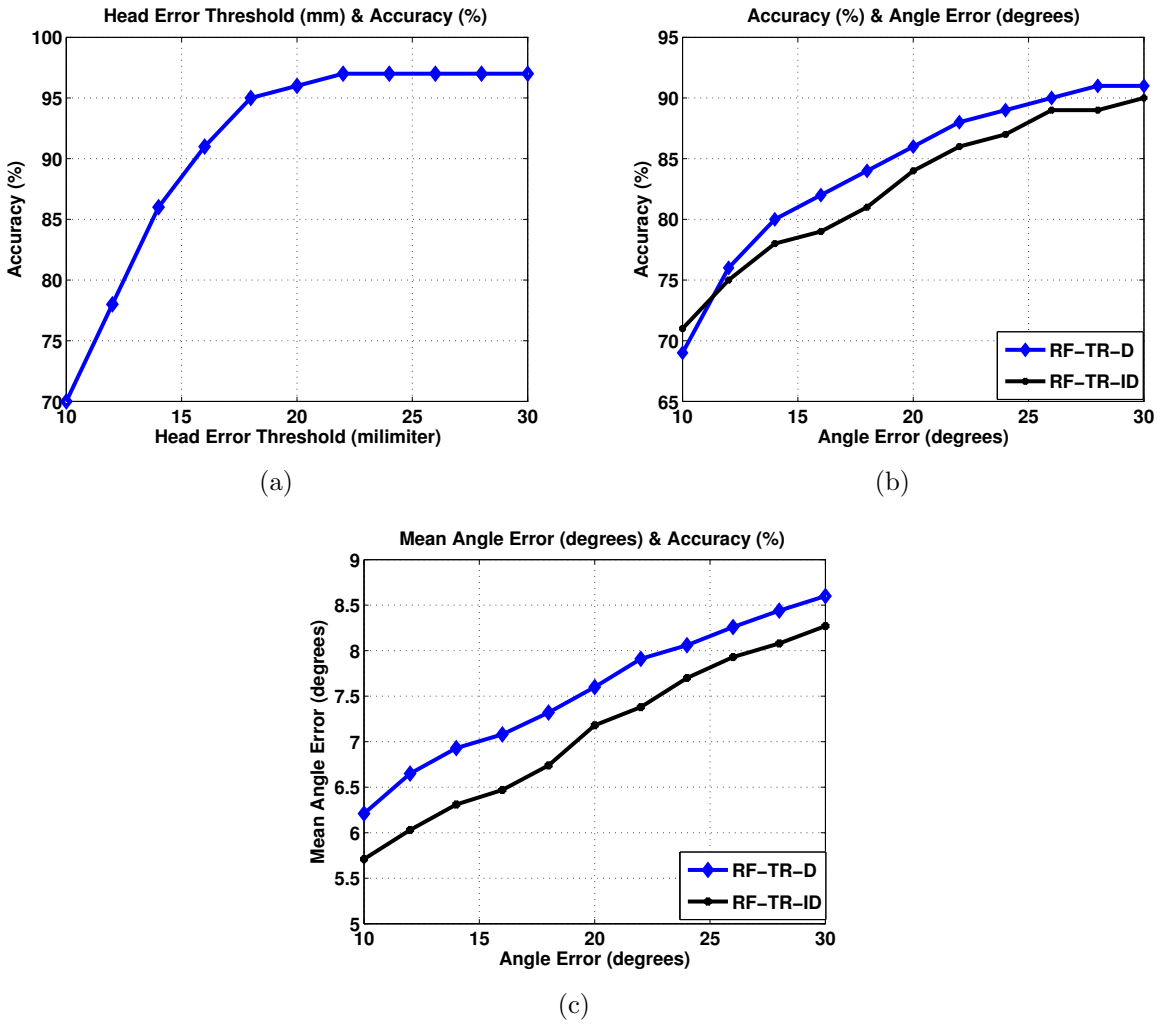



**Figure 4.6:** The performance of the proposed methods, RF-TR-D and RF-TR-ID. (a) Accuracy of head center estimation against different angle thresholds. (b) Accuracy of head orientation estimation against different angle thresholds. (c) Mean Angular Error against different angle thresholds.

slightly more accurate than the RF-TR-ID. However, the proposed method RF-TR-ID resulted in lower mean angle errors for different angle thresholds. (Figure 4.6(c)).

Table 4.1 presents mean and standard errors calculated using 2-fold cross validation. The mean and standard errors are given for RF-TR-D and RF-TR-ID. In Figure 4.7 estimates of head pose parameters using the proposed methods can be seen on the subject's data. The cylinder shows the estimate head pose parameters.

**Table 4.1:** Mean and standard deviation of the head orientation error.

| METHOD | Pitch (∘) | Yaw (∘) | Roll (∘) | Stride |
|---|---|---|---|---|
| **RF-TR-D** | 5.15±0.59 | 7.8±0.70 | 4.8±0.48 | 5 |
| **RF-TR-ID** | 5.08±0.46 | 7.83±0.41 | 5.07±0.19 | 5 |
| **RF-TR-I** | 4.67±0.46 | 8.52±0.41 | 5.57±0.19 | 5 |
| **RF-TR-D** | 5.08±0.14 | 8.2±0.14 | 4.77±0.10 | 10 |
| **RF-TR-ID** | 4.97±0.01 | 8.17±0.18 | 4.49±0.28 | 10 |

**Table 4.2:** Comparison of head estimation results on the Biwi Kinect dataset. We report the Mean Absolute Error.

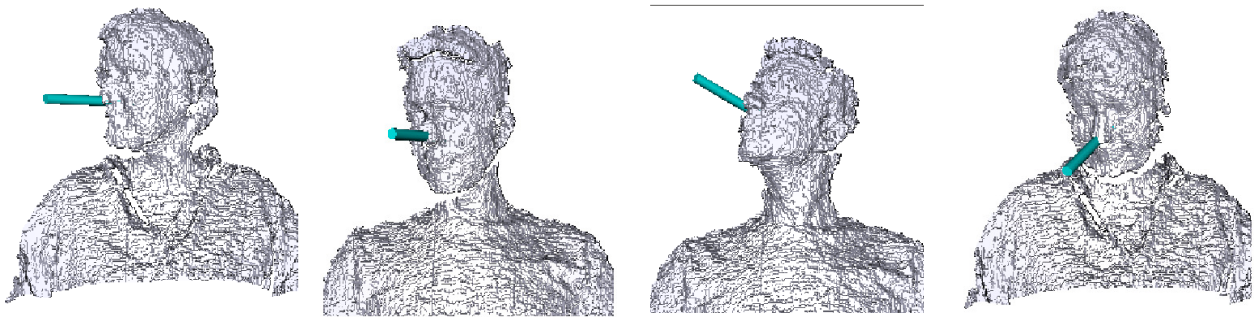
| METHOD | Pitch | Yaw | Roll | Mean |
|---|---|---|---|---|
| **RF-TR-D** | 5.15 | 7.8 | 4.8 | 5.91 |
| **RF-TR-ID** | 5.08 | 7.83 | 5.07 | 5.99 |
| **RF-TR-I** | 4.67 | 8.52 | 5.57 | 6.25 |
| **Random Forests** | 6.71 | 7.95 | 5.67 | 6.78 |
| **Random Forests [18]** | 8.5 | 9.2 | 8.0 | 8.6 |
| **CLM [115]** | 18.30 | 28.30 | 28.49 | 25.21 |
| **CLM-Z [114]** | 12.03 | 14.80 | 23.26 | 16.69 |
| **CLM with GAVAM [114]** | 5.10 | 6.29 | 11.29 | 7.56 |

### 4.4.5 Discussions

The performance of the proposed methods was compared with the performance of the Constrained Local Model (CLM) [115], the CLM that employs both depth and intensity data (CLM-Z) [114] and the CLM-Z with a Generalized Adaptive View-based Appearance Model (GAVAM) [116]. Results are reported in Table 4.2 and in Table 4.3. In Table 4.2, the results for (RF-TR-D and RF-TR-ID) are obtained as the average values

**Table 4.3:** Comparison of head estimation results on the ICT-3DHP dataset. We report the Mean Absolute Error.

| METHOD | Pitch | Yaw | Roll | Mean |
|---|---|---|---|---|
| **RF-TR-D** | 5.85 | 8.53 | 7.64 | 7.34 |
| **RF-TR-ID** | 5.85 | 9.32 | 7.76 | 7.64 |
| **RF-TR-I** | 5.73 | 9.81 | 7.80 | 7.78 |
| **Random Forests** | 6.27 | 9.81 | 7.81 | 7.96 |
| **Random Forests [18]** | 9.40 | 7.17 | 7.53 | 8.03 |
| **GAVAM [116]** | 3.50 | 3.00 | 3.50 | 3.34 |
| **CLM [115]** | 9.92 | 11.10 | 7.30 | 9.44 |
| **CLM-Z [114]** | 7.06 | 6.90 | 10.48 | 8.15 |
| **CLM-Z with GAVAM [114]** | 3.14 | 2.90 | 3.17 | 3.07 |



(a) (b) (c) (d)

**Figure 4.7:** Several examples of head orientation estimation results on the depth data of two subjects from the Biwi Kinect Dataset.

of two runs on the Biwi Dataset. In Table 4.3, the results for (RF-TR-D and RF-TR-ID) are obtained from the ICT-3DHP Dataset. The results of the other methods are reported in [114]. The random forest with Gaussian models is also implemented and the results are provided in the fourth column of Table 4.2 and Table 4.3 for both Biwi and

ICT-3DHP dataset, respectively. We can conclude that the proposed models perform better than the other methods.

As can be seen in Table 4.3 the GAVAM and CLM-Z with GAVAM methods clearly outperform the proposed methods. GAVAM is an integration of differential tracking and keyframe based approaches. This approach is strengthened by using keyframes to avoid drifting. The keyframes depict the face of a certain subject in different poses and scales. These keyframes are acquired and adapted online during testing time. As a tracking method, the performance of the CLM-Z method depends on the initialization of the pose parameters. Once good head pose parameter estimates can be provided by GAVAM, accurate landmark positions could be obtained for the CLM-Z method. Then the tracking provides high precision during parameter estimation. In contrast, the proposed methods estimate the head pose at each frame independently.

The tracking based methods result in accurate parameter estimation on the ICT-3DHP dataset. In this dataset, the subjects move their head slowly in front of a Kinect camera and all frames are captured and stored. However, as can be seen in Table 4.2, the performance of the proposed methods are higher than the tracking based approaches when the Biwi data is used. In this dataset, the subjects move their head fast in front of a Kinect camera. There are also some cases when the frames are lost during the recording of this dataset. To conclude, fast motion and missing frames lead tracking based algorithms to fail. Therefore, the proposed method can be more useful in application where the available data is similar so Biwi dataset, such as that in the Biwi dataset. Estimating accurate head pose on depth data is challenging in the following cases. First, the depth data contains the subject's large head pose. Second, the captured range data might contain some missing data. This causes the extracted patches to be less informative in terms of head orientation. As a result, the estimation accuracy is less accurate.

The pros and cons of tensor model over Gaussian model can be stated as follows. Regression models at the leaf nodes take the data of the test sample into consideration. By contrast, the Gaussian models provide an estimate as the mean of a Gaussian fitted to the training data that arrived at the leaf. That is, they disregard the data/appearance of the test sample (once it arrives at the leaf node of course). There is a (small) computation cost since the parameter prediction using tensor model requires one inner product calculation. On the other hand, the parameter prediction using Gaussian models requires only two 3 dimensional vector additions. The detailed comparisons of the computation times is provided in Figure 4.5

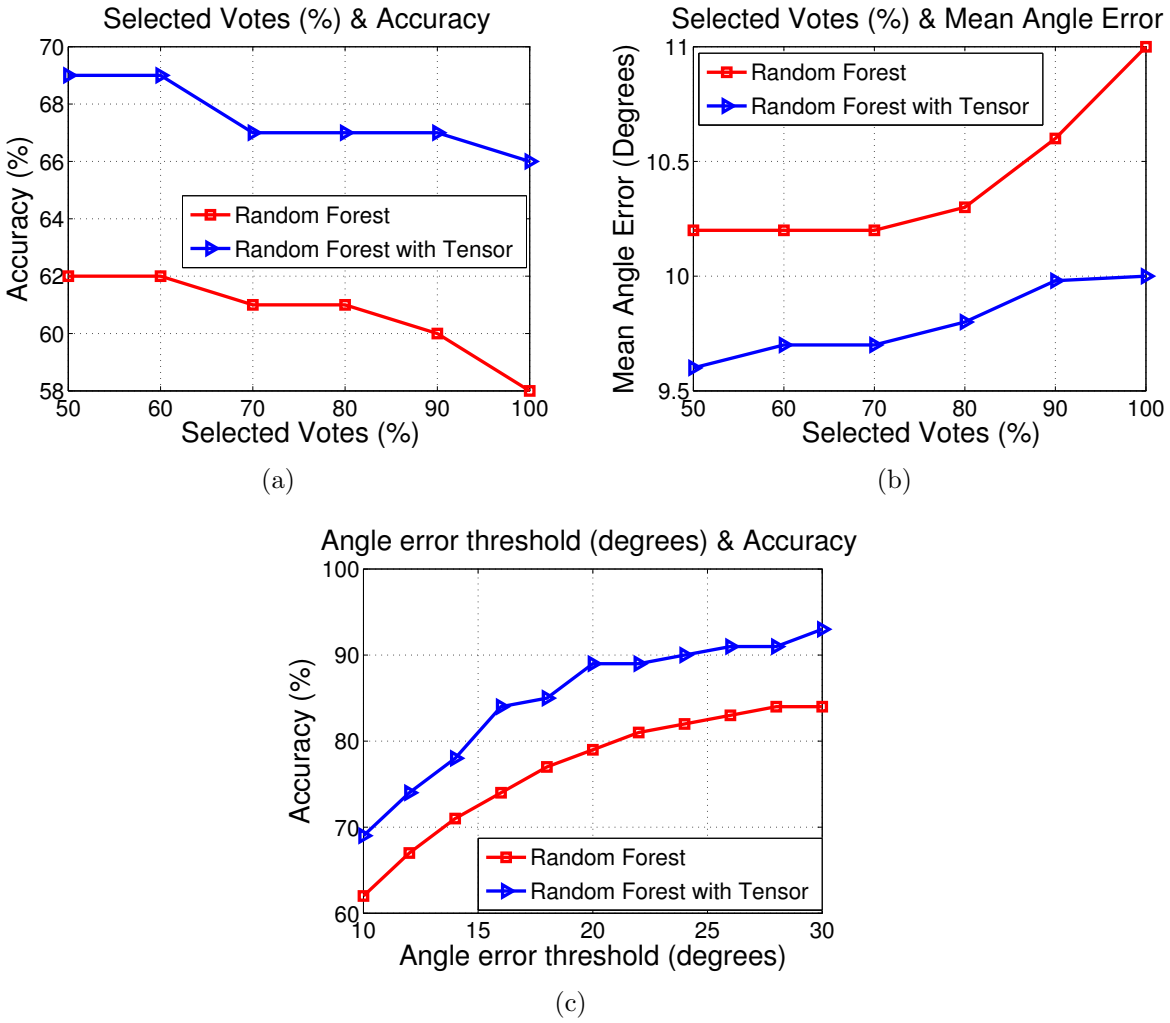
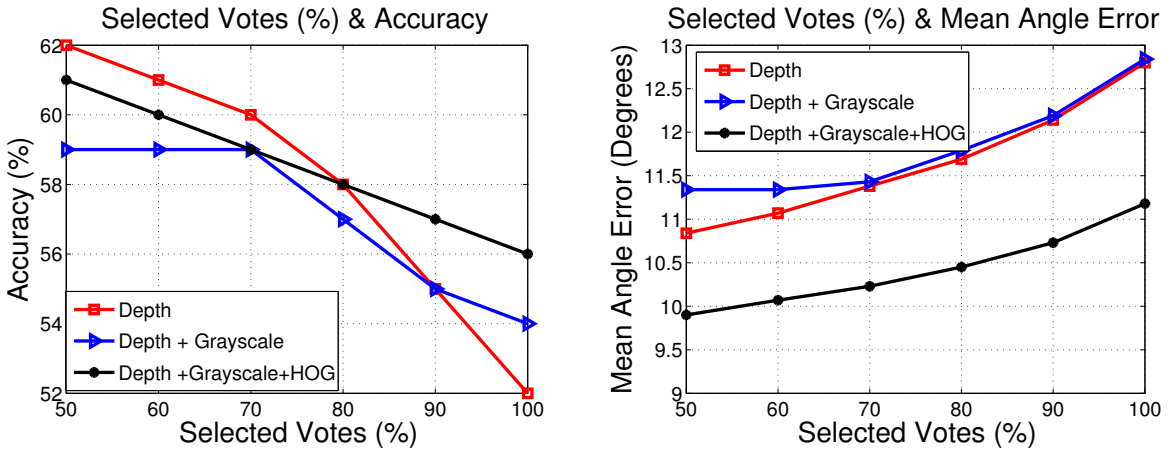



**Figure 4.8:** The performance of typical random forests and random forests combined with tensor regression. (a) Accuracy against the percentage of selected votes. (b) Average angular error against the percentage of retained votes. (c) Accuracy of head orientation estimation for different angle thresholds. The accuracy values were calculated when 50 % votes were selected.

### 4.4.6 Data Fusion using Random Forest

In this section, we study the performance of each type of information (intensity and depth data). More precisely, for intensity data we created ten feature channels, with the first one containing the raw gray values and the remaining 9 channels containing HoG descriptors. For depth data we considered one channel that contained raw depth values.




(a) (b)

**Figure 4.9:** The performance for different types of information. Depth data, depth data combined with gray values and depth data, combined with gray values and HoG descriptors. (a) Accuracy against the percentage of selected votes. (b) Corresponding average angular error against the percentage of retained votes.

The acquired results are reported in Figure 4.8. As can be seen, both higher accuracy and lower MAE were achieved when different sources of information and more than 80 % of the retained votes were used.

Table 4.4 presents mean and standard errors calculated using 2-fold cross validation. In the first row we report mean and standard errors using only depth values while in the second row we report mean and standard errors using depth values, gray scale values and HOG features. A forest is generated using 3000 depth data for the construction of each of seven trees for each fold. As can be seen in Table 4.4, the fusion of different sources of information provided lower mean angular error.

| | **MAE** | **Pitch Error** | **Yaw Error** | **Roll error** |
|---|---|---|---|---|
| **Depth** | 12.7 | 4.8 ± 5.3 | 8.1 ± 9.1 | 6.5 ± 7.4 |
| **Depth+Grayscale+HOG** | 12.3 | 5.0 ± 5.3 | 7.4 ± 7.9 | 6.6 ± 7.4 |

**Table 4.4:** Errors of head orientation in terms of mean and standard deviation. Errors are computed using a 2-fold cross validation.

## 4.5 Conclusion

In this chapter, a novel framework for head pose estimation, called Multimodal Random Forest based Tensor Regression was presented. The techniques were proposed to achieve the head pose estimation using the Kinect. A proposed method allowed employing RGB images and depth data for estimating head pose parameters accurately. The novel method has been developed by extending random forests in three ways: (i) by using tensor-based regression at each leaf node, (ii) by fusing depth and intensity data using tensor regression at each leaf node and (iii) fusing RGB and depth data using random forest. As a result, a new method based on multi-modal random forests and tensor models is proposed for more accurate and robust head pose estimation. The proposed method allows modeling of large head pose while tensor models provide accurate results.

Experimental evaluations of the proposed methods were given in Tables 4.2 and 4.3 respectively. The efficacy of our method was demonstrated on the publicly available Biwi and ICT-3DHP databases. The results proved that the proposed methods have several advantages over classic random forests. First, tensor regressors were trained to map patch data to head pose parameters for more accurate head pose parameters. Second, the proposed method employed both intensity and depth data for more robust and accurate head pose parameter estimation. Furthermore, the efficacy of our method was demonstrated on the publicly available ICT-3DHP databases, Table 4.3. In this dataset, the tracking based methods GAVAM and CLM-Z with GAVAM provided better accuracy than the proposed methods. These methods provided the head pose parameter prediction based on previous prediction parameters. Estimating parameters by making use of previous frames resulted in better accuracy than the proposed methods. If the previous frame is not available or missing during head pose estimation on the current frame, the accuracy of the tracking methods degraded. Since the Biwi dataset contained missing frames during recordings, tracking methods resulted in less accurate results in Table 4.2.

# Chapter 5

# Gaze Estimation

## 5.1 Introduction

This chapter introduces an online learning method for appearance based gaze estimation. The proposed method allows gaze estimation under both frontal and free head pose. Motivation of the proposed online learning is to improve gaze estimation. The state of the art appearance based gaze estimation methods are based on initial personal calibration stage. During this calibration stage, the eye patches of a user are mapped to gaze estimates using a regression model. The proposed online learning method is based on generating models in the calibration stage and then updating these models using both training and testing samples for more accurate gaze estimation. During real-time appearance based gaze estimation, obtaining test eye patches might be different to training samples due to misalignment. This misalignment might result in different eye appearance and the different eye appearance leads to slight changes in actual values of estimates. Furthermore, slight head pose also results in eye appearance variation which affects the gaze estimation accuracy. The proposed online method allows using these samples for gaze estimation in order to improve accuracy. Another advantage of the proposed online method is that when the models are trained for one user, these models can be adapted to another user during online process. Adapting models to another user during online process has several advantages. First, the training samples are substantially decreased. Second, a long calibration time for each subject is avoided.

This chapter also presents appearance based gaze estimation under free head pose from a single device, e.g. a webcam or Kinect. A number of state-of-the art methods are based on several device set-up. It is stated in the introduction section that a system set-

up consisting of several devices is problematic. The proposed method allows estimation of head pose parameters using Kinect depth data and gaze parameters using intensity image. The method also allows head pose parameter estimate from the intensity images.

Performance evaluation of the proposed methods for appearance based gaze estimation are evaluated under slight head pose and free head pose variation. The evaluation of methods are performed using data captured from a singe device, e.g. a webcam or Kinect.

### 5.1.1 Problem Definition

Appearance based gaze estimation is problematic under free head motion. Models trained for gaze direction parameters under frontal head position do not reliably provide accurate estimates under different head translation and orientation. The main reason is that the appearance of eye patches change, therefore models which deal with appearance variation should be considered. The appearance based gaze tracking systems also require a number of training samples and this may lead to a long calibration time. The number of training samples and calibration time increase when these systems are used under free head motion. Therefore, methods which allow fewer training samples are needed to be investigated more.

### 5.1.2 Overview of the Approach

An overview of the proposed method can be seen in Figure 5.1. First, input data, eye patches and head pose parameters, are obtained. The eye corners are detected on intensity image and cropped. The head pose of a user is crucial for appearance based gaze estimation since the pose is used for eye appearance compensation under free head pose. The head pose parameter calculation can be performed in two different ways on the captured intensity images and depth data of the RGB-D camera. Head pose parameter calculations can be carried out on only depth data. The proposed methods are described in the previous chapter. Head pose parameters can also be estimated from estimated facial landmarks. In this case, estimated landmarks and the mean face landmarks are used to estimate the head rotation and translation with respect to mean face. A regression model is learned in order to map eye appearance to gaze direction parameters which allows to obtain gaze point estimates on the screen. In the proposed approach, regression models are generated while the head is in frontal pose. During

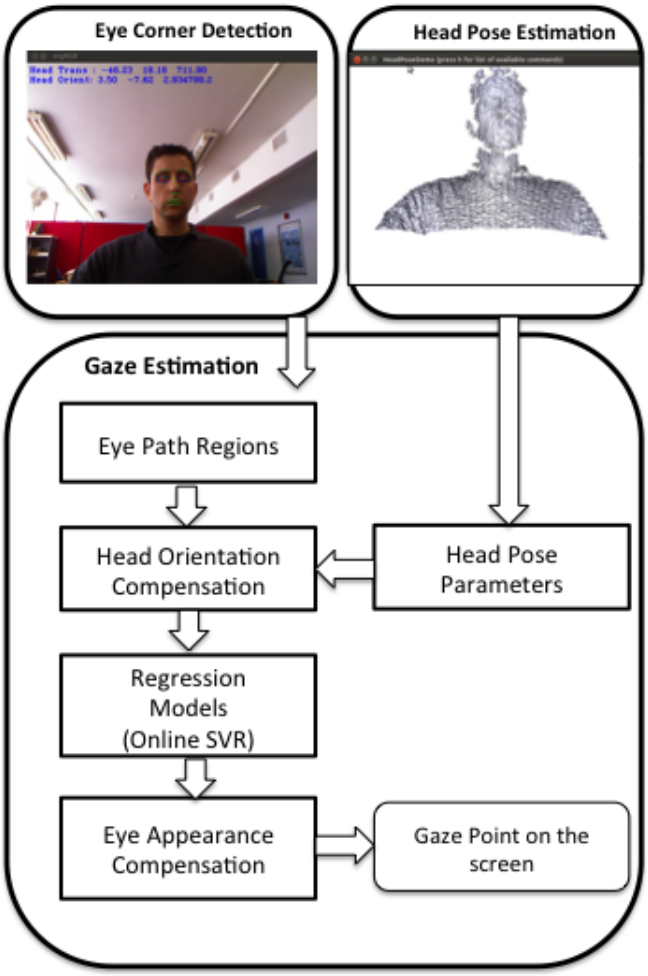



**Figure 5.1:** Proposed method pipeline. a) Landmark points are detected on intensity image. b) Head pose parameters are calculated. c) Eye images are cropped using landmark points. d) Eye gaze parameters are estimated using these cropped eye patches and a regression model. e) Finally, gaze direction parameters are geometrically corrected according to head pose variation.

gaze estimation under free head pose, the captured eye patches are projected to frontal pose and the regression model is used to estimate gaze direction parameters. Estimates are further geometrically corrected to compensate for translation variation. Another recently proposed approach [45] is based on training data collated while the head pose is presented in different directions. However, this approach results in large training data, time and insufficient accuracy. Our approach has the lowest training samples for estimating the gaze direction user free head pose. Furthermore, these estimates are geometrically compensated and final estimates are obtained.

Finally, models are also adapted online using both training and testing eye patches. The model update is based on modeling the training data. Then, the captured test eye patches are selected according to their similarity of training eye patches. The selected eye patches are then used to update the models online. The appearance of the training data is represented using Principal Component Analysis and modeled using a mixture of Gaussian model. The online model update is based on online linear Support Vector Regression (OSVR). OSVR is the online version of the Support Vector Machines (SVR). SVR depend on retraining the models for every new test eye path, however OSVR depends on updating the model for every available eye patch. Therefore, OSVR allows fast computation and real-time model update for gaze estimation. The description of SVR and OSVR can be found in sections 2.4 and 2.5.

## 5.2 Eye Gaze Direction Estimation

In our system set-up, the coordinate system is defined before the mathematical description of the proposed method, Figure 5.2. The World Coordinate System (WCS) is a right-handed 3-D Cartesian coordinate system where XY-plane represents the screen plane. The Kinect camera is placed below the screen and the location of this camera is defined by the camera center or the center of the WCS. A point on the screen is defined by a 3D point, $\mathbf{s} = (s_x, s_y, s_z)$. The center of the eye surface is defined by another 3D point, $\mathbf{c} = (c_x, c_y, c_z)$. $\mathbf{c} = (c_x, c_y, c_z)$ is calculated using the Kinect's camera depth data and facial landmark locations on the intensity image. The visual axis defined by a line which passes from $\mathbf{c} = (c_x, c_y, c_z)$ and intersects the screen plane at $\mathbf{s} = (s_x, s_y, s_z)$. The landmark detection on intensity images is performed using the method proposed in [117]. These estimated points can be seen in Figure 5.3. The corners of left and right eye are selected and the left and right eye patches are determined using these locations on the eye region. The left and right eye center points are also calculated using estimated landmark locations. Figure 5.3 shows detected landmarks (green color), regions of left and right eye patches (blue color) and center points (red color). Several cropped eye patches using this technique are also shown in Figure 5.4. As can be seen in the eye images, the resolution of the images is not very good. Low resolution images result in less accurate gaze direction estimation.

The goal of the proposed approach is to learn a regression function which allows mapping between eye patches $\{\mathbf{e}_1, ., \mathbf{e}_N\}$, and visual axis angles, $\{\delta_1, ., \delta_N\}$ where $\delta =$

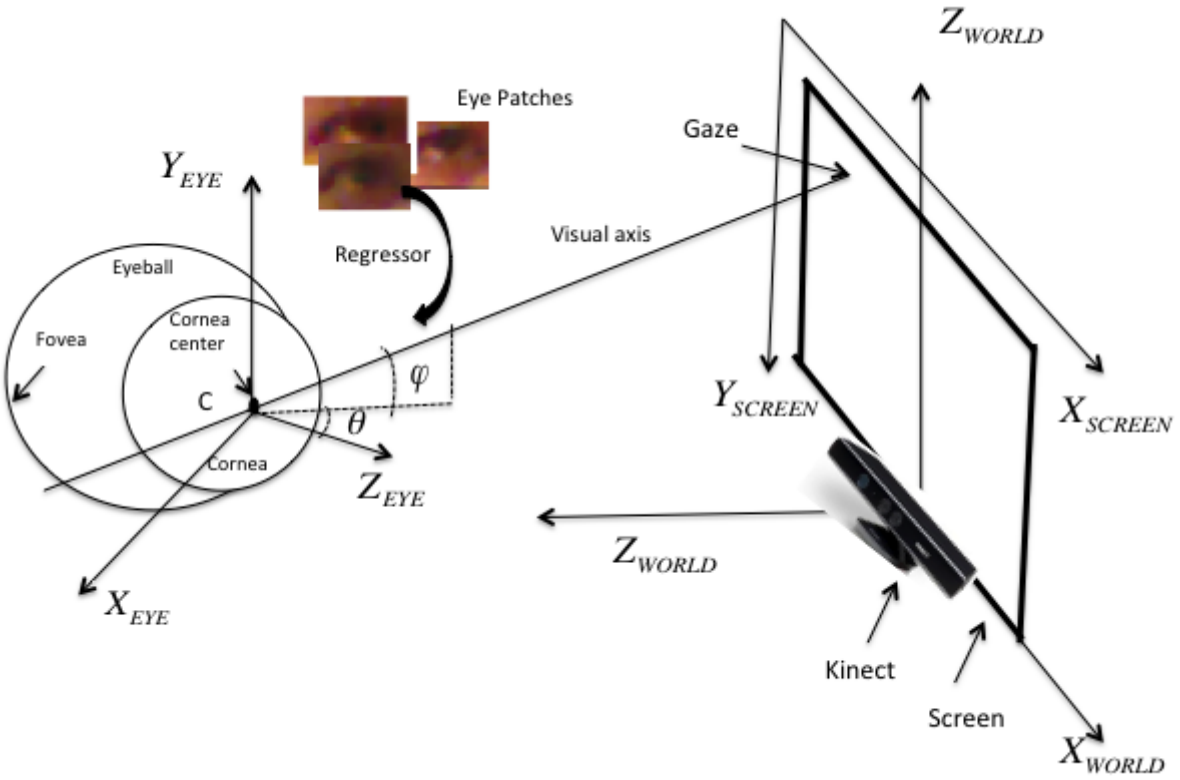
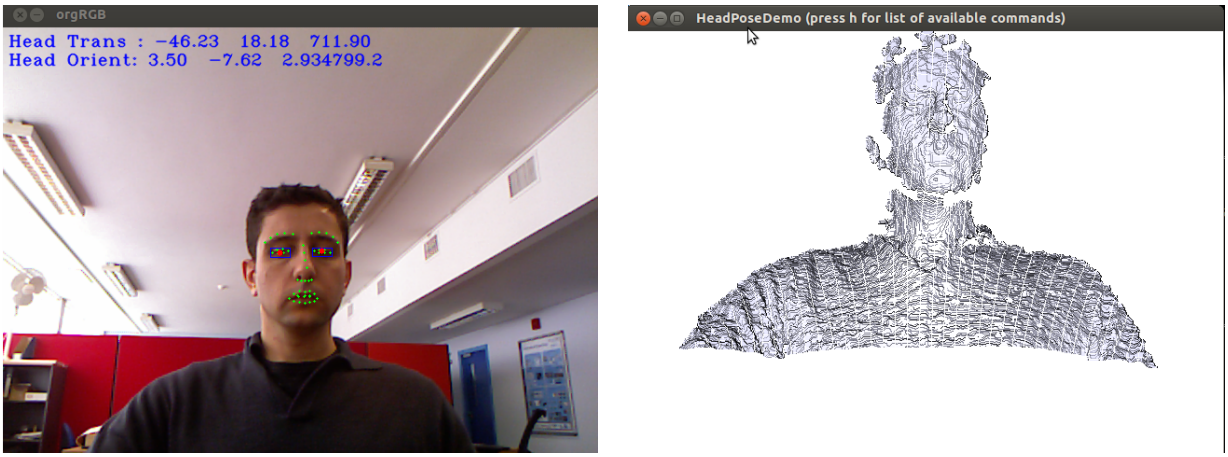


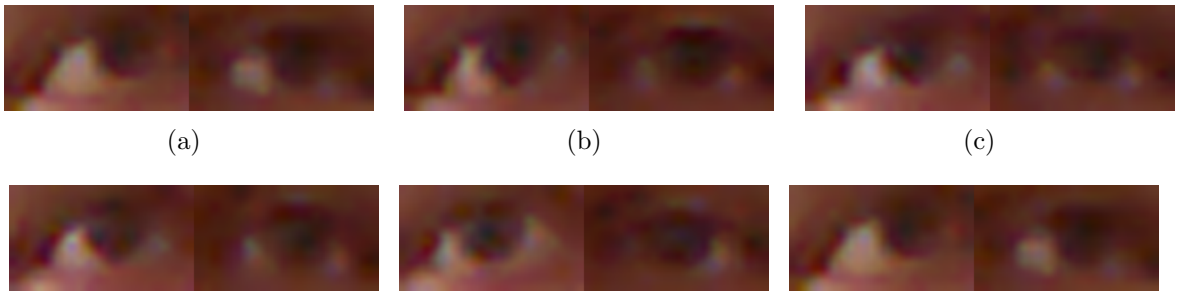

**Figure 5.2:** Mapping eye patches and the orientation of visual axis angles.

$(\theta, \varphi)$ and $\theta$ and $\varphi$ are horizontal and vertical angles of the visual axis respectively. During online gaze estimation, the eye patches are used as inputs to the regression model in order to obtain estimates of visual axis angles, $\dot{\delta} = (\dot{\theta}, \dot{\varphi})$. These angle estimates and the center of the eye patch ($\mathbf{e}$) are then used to obtain the corresponding screen point. The descriptions of the eyeball structure and the visual axis are given in Section 2 Figure 2.2. The section also provides the relation between visual axis parameters and the screen point. However, the estimation of these angels by regression methods provide inaccurate estimates under free head pose. Inaccuracies are expected since the eye appearance of patch varies. The proposed approach, compensates head orientation distortion on the patch by projecting eye appearance, $\mathbf{e}$, to frontal pose, $\hat{\mathbf{e}}$. Compensated eye patch is then used as input to a regression model in order to obtain gaze direction estimates, $\dot{\delta} = (\dot{\theta}, \dot{\varphi})$. These angle estimates and the center of the eye patch ($\mathbf{e}$) are then used to obtain the corresponding screen point. Using this method, the distortion on eye patch then becomes less severe and estimates of angles are more accurate. Although compensating by distortion allows more accurate results, the head translation also results in eye appearance distortion. The compensation of distortion due to head translation




(a) (b)

**Figure 5.3:** Captured intensity image and depth data. a) shows detected landmarks (green color), regions of left and right eye patches (blue color) and center points (red color) b) corresponding aligned depth data.



(a) (b) (c)

(d) (e) (f)

**Figure 5.4:** Several examples of captured left and right eye images using Kinect camera.

is also achieved using geometric correction of angle estimates, $\dot{\delta} = (\dot{\theta}, \dot{\varphi})$. We define another set of angles, $\varphi = (\alpha, \beta)$ which rotates $\dot{\delta} = (\dot{\theta}, \dot{\varphi})$ to the actual/ground truth angles, $\delta = (\theta, \varphi)$. Another regression function is learned to map eye patches to this rotation angles, $\phi = (\alpha, \beta)$. The detailed description is given in Section 5.2.2.

### 5.2.1 Head Orientation Compensation

Eliminating distortion due to head orientation on the appearance of eye patch is achieved in two steps. In the first approach, the perspective projection is used to project the captured eye patch under free head pose to the frontal pose. The estimated head orientation

parameters are used to transfer the pixel values to the frontal pose. The planar perspective projection describes the transformation between two images which share the same optic center, [118, 119]. Figure 5.5 shows the captured image and the projected image. In the second approach, the captured intensity data is mapped onto the captured depth data of the Kinect camera. Then, the texture mapped data is rotated to the frontal position using inverse head orientation parameters. Figure 5.6 shows the captured image and the projected image to the frontal pose.

The first step is carried out as follows. A pair of captured eye patches, $\mathbf{e}$ under free head pose is projected to the frontal pose. Let $\hat{\mathbf{e}}$ be the patch. Then, this pair of patches is used as input to a regression function. Estimates of visual axis angles are also corrected according to the head orientation in order to eliminate the eye appearance distortion caused by head pose changes.

Let $\tilde{\mathbf{p}}$ be the eye image point in the frontal pose (head is not rotated) and $\mathbf{p}$ be the eye image point after the head is rotated. Let $\tilde{\mathbf{\Pi}} = [\tilde{\mathbf{H}} - \tilde{\mathbf{H}}\mathbf{C}]$ and $\mathbf{\Pi} = [\mathbf{H} - \mathbf{H}\mathbf{C}]$ be 3x4 homogeneous projection matrices of $\tilde{\mathbf{p}}$ and $\mathbf{p}$. $\mathbf{C}$ represents the Euclidean position of the camera's center. $\mathbf{H}$ is a 3x3 matrix describing the position and orientation of the eye patch with respect to the world coordinate system. The projection matrix can be decomposed into two parts. First, the camera intrinsic parameters, $\mathbf{K}$, which are calculated during calibration process. Second, the camera extrinsic parameters, $\mathbf{R}$ which describes head rotation around the camera. Euler angles are described by rotation matrices, $\mathbf{R_X}$, $\mathbf{R_Y}$ and $\mathbf{R_Z}$. As a result, the rotation of a head can be described by a rotation matrix $\mathbf{R}$ which is product of three matrices, $\mathbf{R_X}$, $\mathbf{R_Y}$ and $\mathbf{R_Z}$

$$\mathbf{R} = \mathbf{R_X}\mathbf{R_Y}\mathbf{R_Z} \tag{5.1}$$

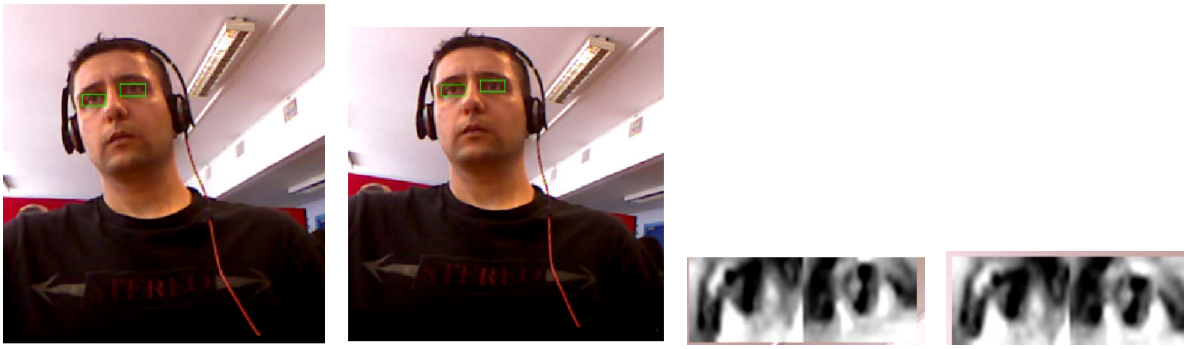
The reprojection of image point $\mathbf{p}$ to frontal pose when the head is rotated is performed using the planar projective transformation which is defined by

$$\tilde{\mathbf{p}} = \tilde{\mathbf{H}}\mathbf{H}^{-1}\mathbf{p} \tag{5.2}$$

$$\tilde{\mathbf{p}} = \tilde{\mathbf{K}}\tilde{\mathbf{R}}\mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{p} \tag{5.3}$$

After this transformation, the reprojected eye image point $\tilde{\mathbf{e}}$ can be obtained. As a result, the head pose appearance variation is eliminated.

The second approach is carried out as follows. The captured depth data is represented as a textured 3D mesh. A 3D mesh is calculated by using depth values of the



(a) (b) (c) (d)

**Figure 5.5:** Description of eye appearence compensation, (a) captured RGB image, (b) frontal RGB image, c) eye patches d) Projected eye patches.

(a) (b) (c) (d)

**Figure 5.6:** Description of eye appearence compensation, (a) captured RGB image, (b) frontal RGB image, c) Cropped eye patches, (d) subregions of eye patches on cropped eye patches.

captured depth data. Then, the textured 3D mesh is rendered to frontal position using the estimated head orientation parameters, $(r_x, r_y, r_z)$. Head orientation parameters are estimated in the World Coordinate System. Figure 5.6(a) shows an example of a captured face image and Figure 5.6(b) shows rotated image to the frontal pose. Figure 5.6 c) shows cropped patches from the rotated image. In Figure d) representing eye patches as a one dimensional vector is described.

### 5.2.2 Head Translation Compensation

Distortion occurs on the appearance of an eye patch while the head moves relative to the camera center. This causes the visual axis angle estimates to deviate from their actual values.

Regression models provide initial estimates of visual axis angles when the head position varies relative to camera. These estimates can be denoted by $\delta^{\mathbf{d}} = (\theta^d, \varphi^d)$. $\delta^{\mathbf{d}}$ represents the distorted angles since these angles are obtained by using distorted eye patched as inputs to regression models. The ground truth of the visual axis angles can be denoted by $\delta = (\theta, \varphi)$ when the head position is in the front of the camera. The bias of the visual axis angles can be denoted by $\Delta\mathbf{\Omega} = [\delta^{\mathbf{d}} - \delta] = [(\theta^d - \theta), \varphi^d - \varphi)]$. The bias, $\Delta\mathbf{\Omega}$, is horizontal and vertical rotation angles which rotates distorted visual axis angles, $\delta^{\mathbf{d}}$ to actual values, $\delta$. We trained a regression model to map left and right eye patches to this bias. During testing, the estimates of visual axis angles, $\dot{\delta} = (\dot{\theta}, \dot{\varphi})$ are obtained and the learned bias is used to rotate the distorted visual axis angles to actual values. Finally, the point on the screen is obtained using these estimates and the distance from screen.

The parameter bias is learned during the training and used to compensate the bias during the testing. Learning of biases is performed by recording a short video clip. A number of points are displayed at random locations on the screen. A user is instructed to look at the point and move his/her head while looking at the point. Estimates of visual axis angles, $\delta^d$, are calculated and then bias is calculated as described above. Finally, a regression function is learned to map an eye patch to the bias. The performance evaluation of estimates of the translation bias obtained from a regression model is compared with ground truth and results are given in the experimental section.

## 5.3 Adaptive Eye Gaze Direction Estimation

The method proposed above is further extended to become an online learning method for appearance based gaze estimation. The method is based on the eigenspace representation of the eye patches. During the initial calibration state, the eye patches are represented with principal component features. Then, these features are mapped to screen points using a regression function, namely online SVR that is trained on principal component features. The probability density of the features is also represented using

a Gaussian Mixture model. During online gaze estimation process, the test eye patches are captured and the likelihood that they are sampled from the same distribution as the training samples is calculated using the Gaussian mixture model. The eye patches with sufficiently high likelihood are then selected in order to update both the eigenspace representation and the regression models for left and right eye. Figure 5.7 describes the online learning and adaptation engine for the appearance based gaze estimation.

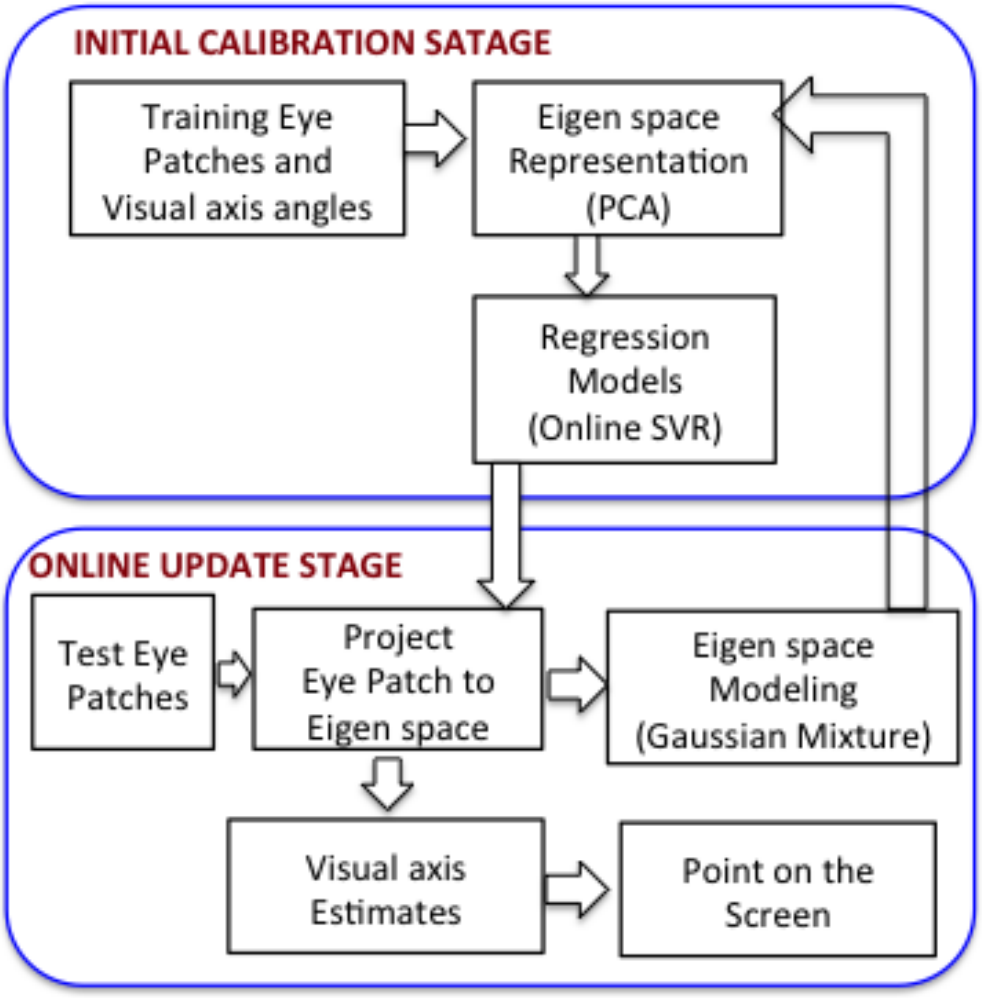



**Figure 5.7:** Initial calibration and online update stages. Initial calibration stage involves generation of the eigenspace and the regression models. Online update stage involves updating both the eigenspace and regression models.

### 5.3.1 Training Data Modelling

The initial calibration stage is performed as follows. Capturing a set of eye patches of a user is carried out by instructing him/her to follow a set of points on the screen during the recording. Given a set of eye patches, $\{\mathbf{e}^t\}_{t=1}^N$, projections of these eye patches, $\{\hat{\mathbf{e}}^t\}_{t=1}^N$, are generated as described in Section 5.2.1. Given a set of projected eye patches, $\{\hat{\mathbf{e}}^t\}_{t=1}^N$, training vectors $\{\mathbf{x}^t\}_{t=1}^N$, are generated. $\mathbf{x}$ contains the pixel values of $\hat{\mathbf{e}}$ in one column. Given the set of vectors, $\{\mathbf{x}^t\}_{t=1}^N$, we calculate corresponding principal component feature vectors by solving the eigenvalue problem. As a result, we obtain $\mathbf{y} = \{y_1, ...., y_N\}$ and the corresponding $\{\delta_1, ., \delta_N\}$.

After, online SVR is trained in order to map projection vectors $\mathbf{y} = \{y_1, ...., y_N\}$, to visual axis angles, $\{\delta_1, ...\delta_N\}$. Then, the visual axis horizontal and vertical angles can be obtained using the linear SVR as follows:

$$f(\mathbf{y}; \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{y} + b, \tag{5.4}$$

where $\mathbf{y}$ is the projection vector, $\mathbf{w}$ is the weight vector, and b is the bias. The output $f(\mathbf{y}; \mathbf{w}, b)$ is the estimate of a horizontal or vertical angle. In our study, we generate two regression models. One of models allows mapping projection vectors to horizontal and another one allows mapping projection vectors to vertical angles of the visual axis.

Our proposed online learning method is based on capturing a test eye patch and associated visual axis angles $(\mathbf{y}^*, \delta^*)$ and then updating online SVR models online. However visual angles of captured test eye patch is not known. We achieve estimating these angles by modelling the projection vectors using a Gaussian mixture model during training. A number of screen points can be denoted by K and a number of associated projection vectors for each screen point can be denoted by J. We also denote jth projection vector of each eye patch corresponding to kth screen point by $y_{jk}$. Since there are K sets of projection vectors, we can model densities of sets $P_1(y), ....P_K(y)$ using a parametric mixture model. The model is defined by

$$P(\mathbf{y}|\Theta) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{y}|\mu_i, \Sigma_i) \tag{5.5}$$

where $\mu$ is mean vector, $\Sigma$ is covariance matrix and $\alpha$ is a mixing parameter of a component, $\sum \alpha_i = 1$. The mixture can be defined by a set of parameters, $\Theta = \{\mu_{1,....M}, \Sigma_{1....M}, \alpha_{1....M}\}$.

These unknown parameters $\Theta = \{\mu_{1,....M}, \Sigma_{1....M}, \alpha_{1....M}\}$ need to be estimated. Our approach is to maximize the joint likelihood $P(\mathbf{y}|\Theta)$ of projection vectors $\{\mathbf{y}^t\}_1^N$

$$\mathbf{\Theta}^* = \arg\max\left[\prod_{t=1}^{N} P(\mathbf{y}^t|\Theta)\right] \tag{5.6}$$

Parameter estimation can be achieved using EM algorithm.

After generating the Gaussian Mixture model, we create M sets of visual axis parameters and calculate the average values for each set $\{\hat{\delta}_j = \hat{\theta}_j, \hat{\varphi}_j\}_{j=1}^M$. For each $\mathbf{y}_j$, the density estimate $P(\mathbf{y}|\Theta)$ of projection vectors can be used to fine the representative set by calculating

$$\mathcal{S}_j = \text{argmax} \quad p(j|\mathbf{y}) \tag{5.7}$$

where $\mathcal{S}_j$ denotes $j^{th}$ class and $\mathbf{y}$ is the projection vector. $p(j|\mathbf{y})$ denotes the posterior probability of the $j^{th}$ component of the Gaussian mixture component. In this way, $j^{th}$ set is chosen for the given projection vector and the corresponding visual axis angles are stored in this set.

During testing, projection vector $\mathbf{y}^*$ is used to calculate posterior probability to find the most probable mixture component (jth component). The mean value of visual axis angles of jth component $\{\hat{\delta}_j = \hat{\theta}_j, \hat{\varphi}_j\}$ is used as an estimate visual axis angles of captured test eye patch.

### 5.3.2 Maximum Likelihood Detection

In this section, updating the models online will be described. Experiments were conducted using all available test eye patches, however the accuracy did not improve the parameter estimation. Removing outliers is crucial. Outliers might occur when capturing a eye image of the corresponding calibration point, if the user is not looking at that point. Outliers are also produced due to eye blinks. The eye blink results in occlusion of the eye. For this reason, the proposed method selects test eye patches for model update. This section includes several steps for online update of the regression models. First, selecting a test eye patch for model update is described. Second, calculation of the visual axis angle of the selected eye test patch is described. Finally, updating the projection vectors and the online SVR are given.

**Algorithm 2 Online adaptive appearance based gaze estimation.**

**Representation**: Projection vectors $\mathbf{y} = \{y_1, ...., y_N\}$ are calculated using PCA.

**Modelling**: The projection vectors, $\mathbf{y} = \{y_1, ...., y_N\}$, are modelled by a Gaussian mixture model.

**Learning**: Learning a regression model which maps projection vectors, $\mathbf{y} = \{y_1, ...., y_N\}$, to visual axis angles, $\{(\delta_\mathbf{1}), ., (\delta_\mathbf{N})\}$.

**Clustering**: Group projection vectors according to mixture components.

**Detecting test eye patches and Updating online SVR model**:

**for** each new eye patch **do**

Determine the most likely Gaussian mixture component of a test projection vector. In other words, calculate a posteriori probabilities of each mixture component and select the one which provides maximum value.

**if** Maximum a posteriori probability $> \tau_\mathbf{1}$ and $H(S) > \tau_\mathbf{2}$ **then**

Add test projection vector to the training set,

Calculate corresponding visual axis rotation of this projection vector and add to the training set

Update PCA coefficients and

Retrain the online linear SVR

**end if**

Provide estimates of visual axis angles

**end for**

Selection of a test eye patch online is carried out as follows. The eye patch is projected to eigenspace. Then, the most probable mixture component of this projection vector is calculated. Determining the most probable mixture component is described in Section **??** and the posterior probability calculation is described by Equation 3.37. After, a threshold is applied to this probability. Eye patches which have high likelihood value (higher posterior probability) are selected. The threshold is selected using experimental observations and denoted by $\tau_1$. The eye patches which have low likelihood values are rejected. The corresponding visual axis angles, $\dot{\delta}$, of the selected eye patches are also calculated online. The corresponding class of the most probable mixture component is determined. Let $S_c$ represent a set of visual axis parameters of class c. Then, we model the distribution of the set of visual axis parameters as a multivariate Gaussian,

$$p(\delta) = \mathcal{N}(\delta; \bar{\delta_c}, \Sigma_c) \tag{5.8}$$

The differential entropy of this distribution is defined by

$$H(S) = \frac{n}{s}(1 + \log(2\pi)) + \frac{n}{2} log|\Sigma_c| \tag{5.9}$$

The test eye patch is then selected and the corresponding cluster is determined, the corresponding visual axis angles are calculated by averaging the visual axis angles which are in the cluster. The accuracy of the average value depends on the value of $H(S)$. The value of $H(S)$ determines the similarity of visual axis angles in the class. A high value of $H(S)$ shows that the angles in a class are very similar in the cluster. Another threshold is applied to $H(S)$ in order to select or reject an eye test patch online. This threshold is denoted by $\tau_2$. The threshold is selected using experimental observations. In this way, the test eye patches are included in the training set which are different in appearance than the other training eye patch. This allows us to better model the eye appearance.

Finally, using the training samples and the selected test samples, a new $\hat{\mathbf{\Phi}}$ is calculated. Similarly, new projections are calculated by $\tilde{\mathbf{y}} = \tilde{\mathbf{\Phi}}^T \hat{\mathbf{x}}$. Finally, these projected vectors are used as inputs to a online linear SVR to obtain more accurate estimates.

The performance evaluation the proposed method is done by comparing parameter estimation using only training samples and parameter estimation using both training and testing samples. Higher accuracy is achieved by using both training and testing samples.

## 5.4 Experimental Results

In this section, the evaluation of the proposed method was performed by conducting a number of experiments. The system set-up was done using a desktop PC and a Kinect camera. A 22-inch LCD monitor was positioned about 70cm from the subject and the Kinect camera placed under the screen.

In this section, the performance evaluations of the proposed methods for the appearance based gaze tracking are evaluated. In section 5.4.1, the description of the dataset creation using Kinect camera is described. Then, the performance comparison of the commercial and the proposed appearance based gaze tracker is described in section 5.4.2. After, the experimental results for gaze estimates under slight head motion are reported in section 5.4.3. Moreover, the eye appearance variation due to head translation is ad-

dressed in section 5.4.4. Furthermore, in section 5.4.5, the experimental results for gaze estimates under free head pose head motion are reported. In this section, two different evaluations are provided. First, the accuracy of gaze estimates is evaluated using the head pose parameters which are calculated using Kinect depth data. Second, the accuracy of gaze estimates is evaluated using the head pose parameters which are calculated using detected landmarks detection on intensity image. Finally, the proposed online learning method for appearance based gaze estimation for person independent gaze tracking is introduced in section 5.4.7.

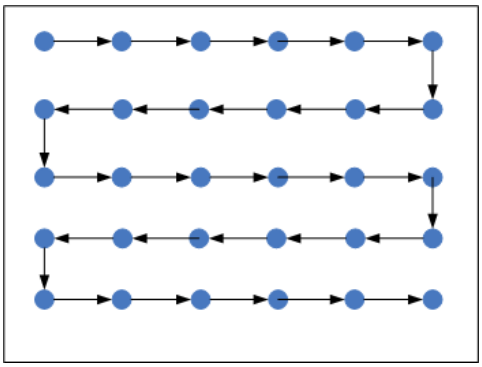


**Figure 5.8:** Calibration points which were displayed on the screen. The arrow shows the order of the displayed points.

### 5.4.1 Data Collection

Training and two testing data were collected using the system set-up described above. The chin rest has not been used during recordings since the aim is to develop a system for practical applications. Figure 5.8 shows locations of 30 calibration points on screen. These points were shown to each subject while training and testing sets were collected. Figure 5.9 also shows examples of captured intensity images and depth data of subjects involved in the experiments.

1. **Training data collation under slight head motion**. A subject was shown 30 points(6x5) sequentially on the display. While the subject was looking at these points, his/her intensity images and depth data are captured.

2. **Testing data collation under slight head motion**. A subject was shown 100 random points on the display. While the subject was looking at these points under slight head motion, his/her intensity images and depth data are captured.

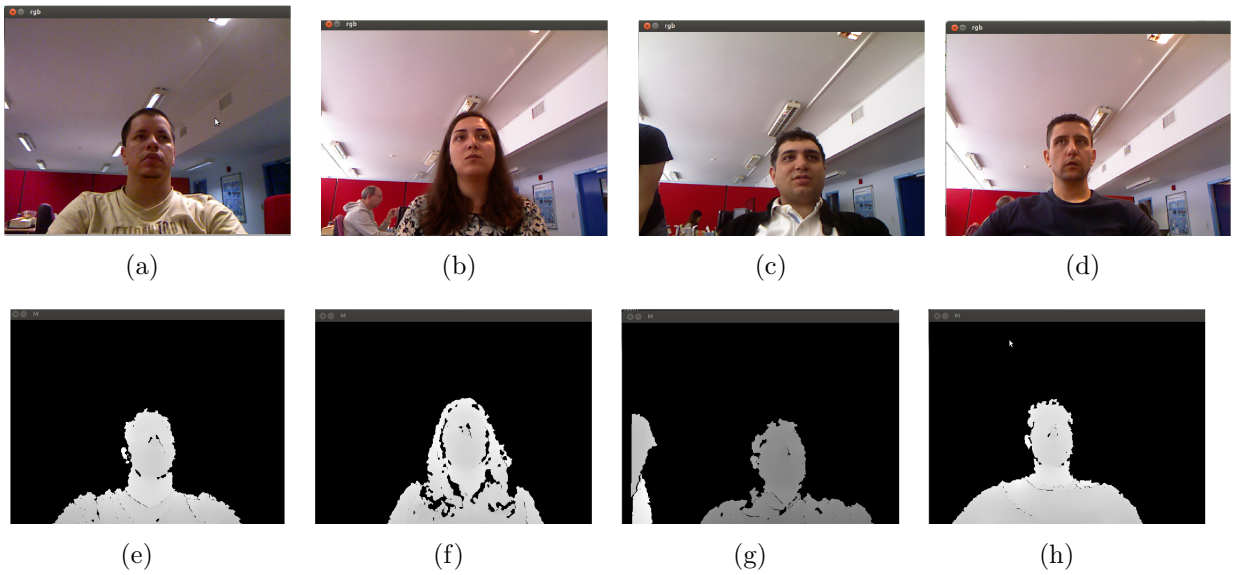



**Figure 5.9:** Examples of captured intensity images and depth data of subjects who involved in the experiments.

3. **Testing data collation under free head motion**. A subject was shown 100 random points on the display. While the subject was looking at these points under free head motion, his/her intensity images and depth data are captured.

### 5.4.2 Comparison of commercial gaze estimator and the proposed method

The performance comparison of a commercial gaze estimator and the proposed method is provided in this section. The system set-up was done using a desktop PC and a Kinect camera. The 17-inch LCD monitor was positioned 70cm from the subject and the Kinect camera placed under the screen. The commercial gaze estimator ran in parallel with the proposed method. The reference points 6x4 are shown to a user while training and testing data are collected using Kinect. Leave-one-out experiments were conducted to obtain the average error values of the method. The average errors of the consumer tracker and the proposed method are 0.9 degrees and 1.8 degrees respectively.

| **Method** | **Error** | Training Samples |
|---|---|---|
| **Proposed** (Grayscale) | **1.8** | **30** |
| Lu et al. [45] | 0.85 | 33 |
| S3GP+edge+filter [42] | 0.83 | 16 label and 75 unlabled |
| Tan et al. [26] | 0.5 | 252 |
| Baluja et al. [40] | 1.5 | 2000 |
| Xu et al. [41] | 1.5 | 3000 |

**Table 5.1:** Comparison of gaze estimation performance under slight head motion. Leave one results were reported in the table.

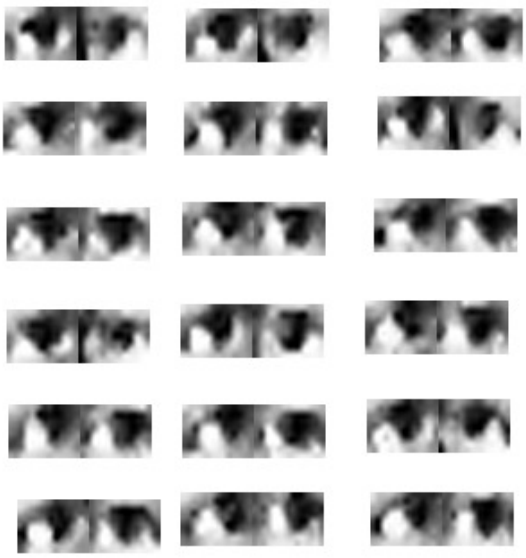
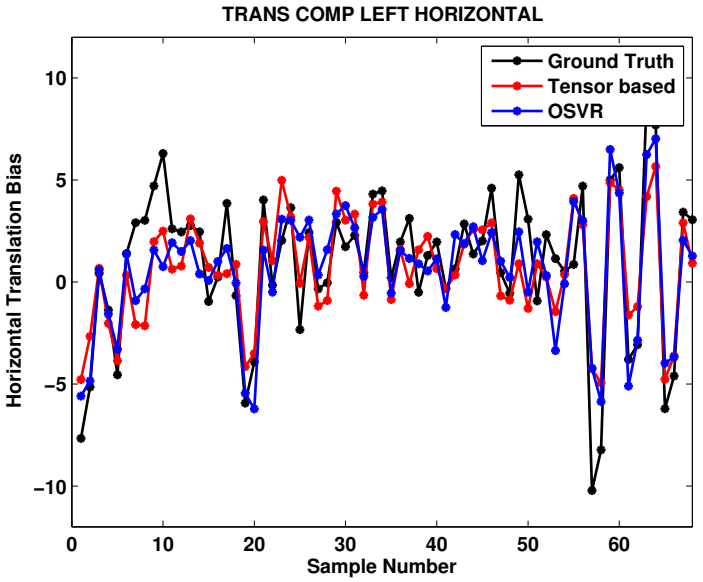


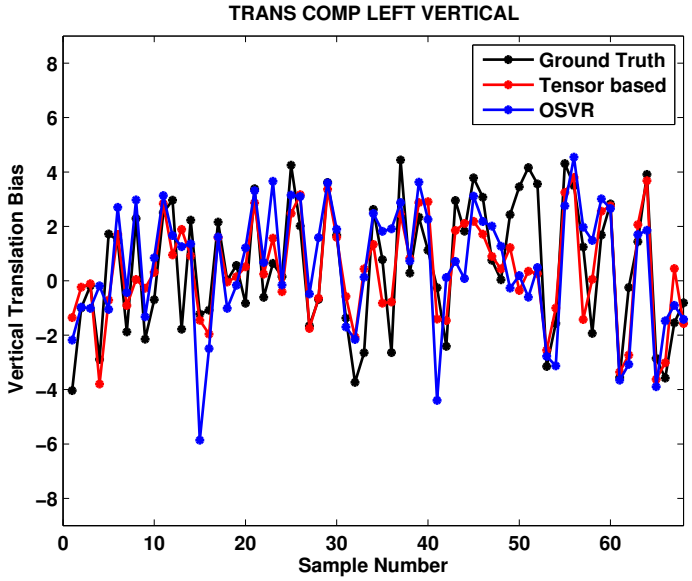
**Figure 5.10:** Distorted eye images due to translation variation.

### 5.4.3 Gaze Estimation under slight head motion

The performance of the proposed method was also evaluated by performing leave-one-out experiments. The gaze estimation results were compared with other methods in Table 5.1. The obtained accuracy is comparable to other appearance based methods, [26,40–42,45]. Regression models were trained using a small number of training samples.




(a)




(b)

**Figure 5.11:** Appearance distortion compensation for the second system. Estimates of translation angles and the actual angles are reported in the figure to evaluation of the accuracy. a)Horizontal and b)vertical translation angle estimates obtained from regression models.

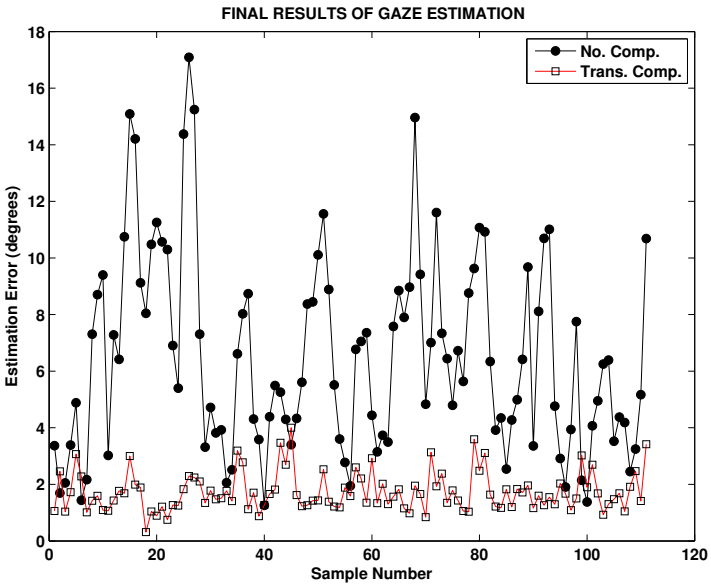



**Figure 5.12:** The experimental evaluation of with/without appearance distortion compensation.

### 5.4.4 Gaze Estimation under eye appearance variation

The proposed appearance distortion method in Section 5.2.2 was validated by conducting experiments. The user looked at a fixed point on the screen and move his head while looking at the point. Eye patches were captured together with translation angles (horizontal and vertical). The eye patches and corresponding translation angles were used to train regression models. One regression model was trained for horizontal translation angle and another one was trained for vertical translation angle. In Figure, 5.10, shows a number of eyes which were captured under translation variation. In Figure, 5.11, the estimates of translation angles which were obtained from regression models were given.

The performance of the appearance compensation method was evaluated by performing leave-one-out experiments. Initial estimates were obtained from tensor models which were trained under fix head pose and then estimated translation angles were used for compensation. The average angular errors with and without translation variation compensation were calculated as 6.5°, 1.7° respectively. In Figure 5.12, we present the experimental evaluation with/without appearance distortion compensation. The proposed method for translation variation compensation results in higher accuracy.

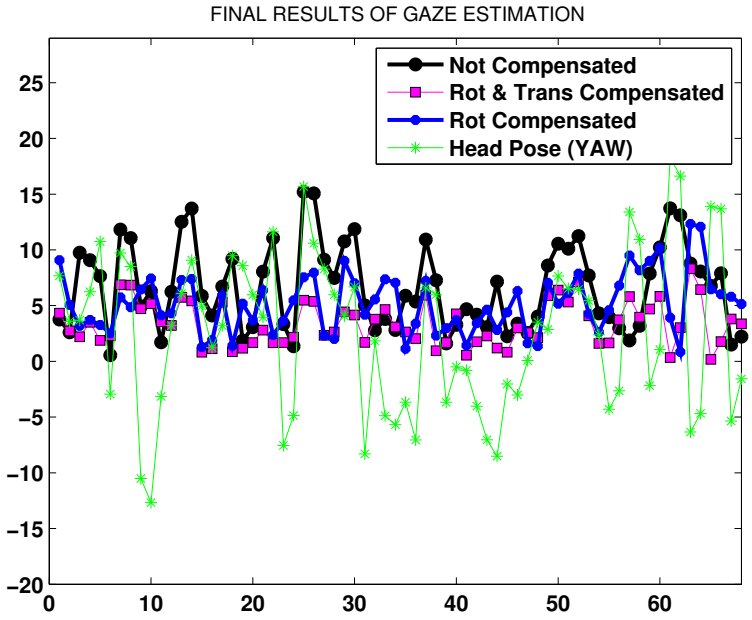



**Figure 5.13:** Performance of appearance based gaze estimation under free head pose. Angular error values are reported in degrees for different proposed compensation methods. The head pose parameters are calculated using detected landmarks on intensity image.

| Range | Value |
|---|---|
| X-trans. | $-244 \sim 157$ mm |
| Y-trans. | $-11 \sim 143$ mm |
| Z-trans. | $489 \sim 867$ mm |
| Pitch | $-23° \sim 10°$ |
| Yaw | $-21° \sim 22°$ |
| Roll | $-18° \sim 22°$ |

**Table 5.2:** Head pose Values and Ranges.

### 5.4.5 Gaze Estimation under free head pose

Four subjects were involved in the experiments in order to test the gaze direction estimation under free head motion. Data collection was carried out as described in Section 5.4.1.

The overall performance evaluation is given in Table 5.3 showing rotation and distortion compensation results in higher accuracy. Table 5.3 also provides results of the previous studies. Figure 5.13(a) shows with/without compensation results. The four plots are reported for the proposed compensation methods. As can be seen on the plots, the rotation and translation compensation provides higher success rate for the gaze estimates. The range of translation and rotation values can be seen in Table 5.2

| METHOD | Rot. and Dist. Comp. | Dist. Comp. | Rot. Comp. | No Comp. | Training Samples |
|---|---|---|---|---|---|
| **Subject 1** | 4.05 | 10.42 | 7.88 | 8.72 | 133 training samples |
| **Subject 2** | 2.86 | 8.21 | 7.42 | 8.66 | |
| **Subject 3** | 4.12 | 9.70 | 7.42 | 8.21 | |
| **Subject 4** | 2.90 | 6.31 | 6.17 | 6.21 | |
| **Average** | **3.40** | | | | |
| Lu et al. [45, 120] | $\sim 3$ | | | | 33 training samples and 5-second video clip |
| Sugano et al. [43] | $4 \sim 5$ | | | | $10^3$ |

**Table 5.3:** The performance evaluation of the proposed method under free head motion. Average mean errors and mean angular errors for four subject, measured in degrees. The head pose is calculated using landmark points.

Furthermore, the percentages of correct gaze estimates as a function of success are reported in Figure 5.14. Lower error percentages with higher percentages occur when rotation and translation compensation has been employed. The results shows that only translation or orientation compensation does not provide accurate results.

The histograms of errors (millimeter and angular) is reported in Figue 5.16.

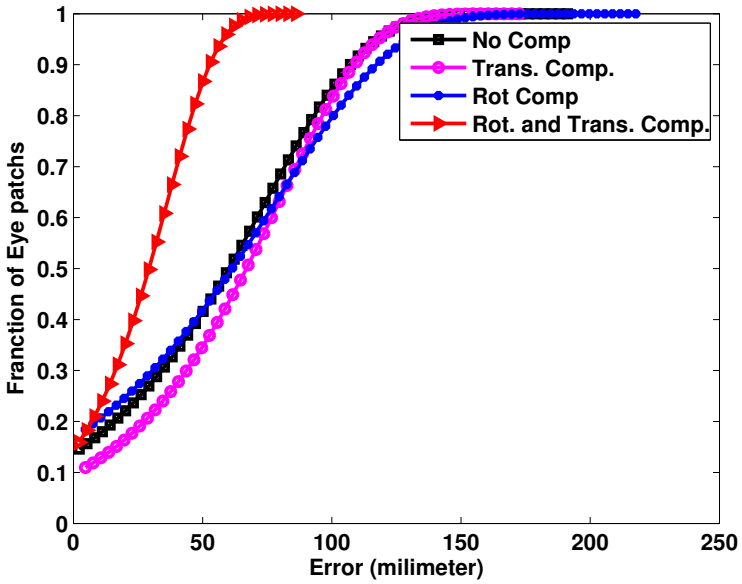
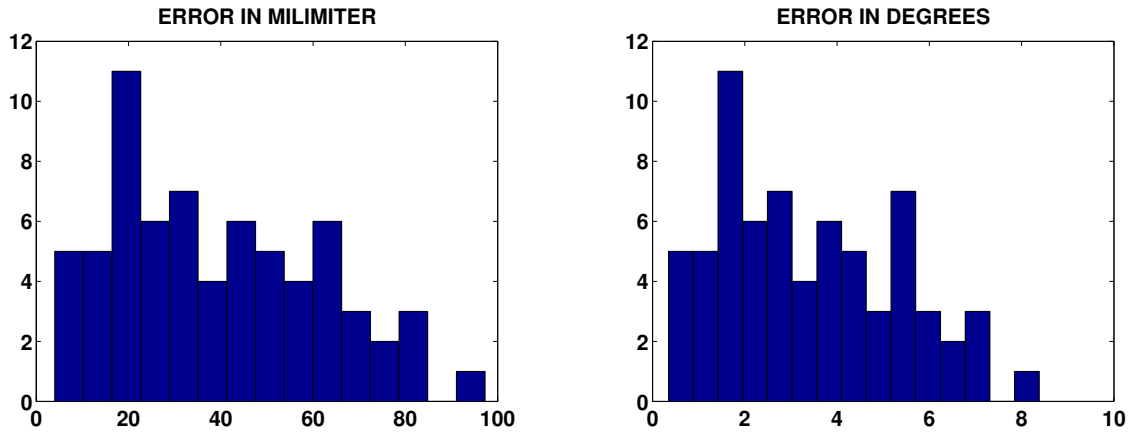



**Figure 5.14:** The accuracy of the proposed methods. The thresholds define the success. Average results are reported in degrees.

| METHOD | Rot. and Dist. Comp. | Dist. Comp. | Rot. Comp. | No Comp. | Training Samples |
|---|---|---|---|---|---|
| **Subject 1** | 5.17 | 13.0 | 8.2 | 7.6 | 133 training samples |
| **Subject 2** | 4.86 | 15.0 | 7.0 | 4.6 | |
| **Subject 3** | 3.25 | 10.2 | 6.4 | 6.05 | |
| **Subject 4** | 4.9 | 10.7 | 7.58 | 7.33 | |
| **Average** | **4.5** | | | | |
| Lu et al. [45, 120] | $\sim 3$ | | | | 33 training samples and 5-second video clip |
| Sugano et al. [43] | $4 \sim 5$ | | | | $10^3$ |

**Table 5.4:** The performance evaluation of the proposed method under free head motion. Average mean errors and mean angular errors for four subject, measured in degrees. The head pose is calculated using Kinect depth data.

In the second experiment, the head pose parameter estimation was performed on the captured depth from Kinect camera and then the estimated parameters were used to eliminate eye appearance distortion on eye images. The overall performance evaluation is given in Table 5.4 showing rotation and distortion compensated results in higher accuracy. Table 5.4 also provides results of the previous studies. Figure 5.15 shows with/without compensation results. The Table 5.5 also shows the head pose range of a subject involved in the experiment.




(a) (b)

**Figure 5.16:** Gaze estimation under free head pose. (a) Histogram of distance error in millimeter. (b) Histogram of angle error in degrees.

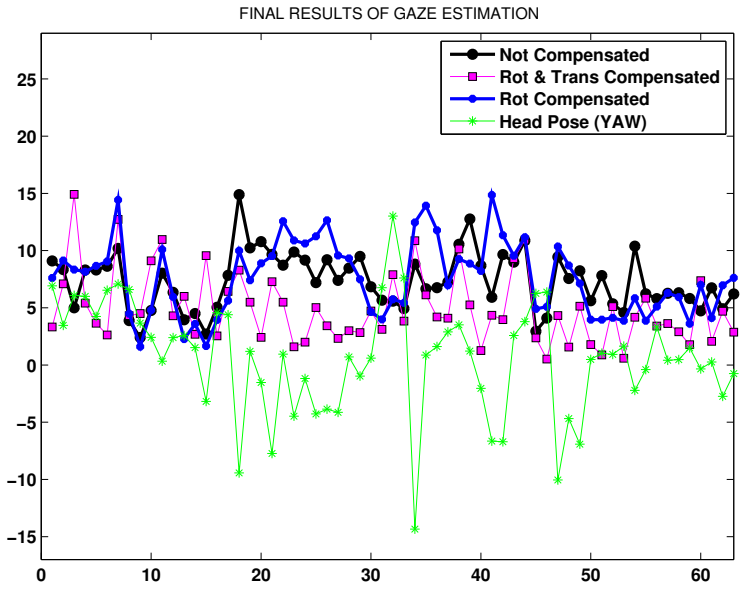



**Figure 5.15:** Performance of appearance based gaze estimation under free head pose. Angular error values are reported in degrees for different proposed compensation methods. The head pose parameters are calculated using Kinect depth data.

| Range | Value |
|---|---|
| X-trans. | $-244 \sim 157$ mm |
| Y-trans. | $-11 \sim 143$ mm |
| Z-trans. | $489 \sim 867$ mm |
| Pitch | $-23° \sim 10°$ |
| Yaw | $-21° \sim 22°$ |
| Roll | $-18° \sim 22°$ |

**Table 5.5:** Head pose Values and Ranges.

### 5.4.6 Online Gaze estimation

A performance comparison of the performance of the model learned in the initial calibration stage and the model that is learning gaze estimates online is given in Figure 5.17. In particular, regression models which were generated during initial calibration stage were updated using test eye patches. These test eye patches were selected among available test eye patches. Updating the regression models online using selected test eye patches results in higher accuracy. Figure 5.17 provides total error of gaze estimates in degrees for every 20 frames. There is a clear improvement in gaze direction estimation errors. As it can be seen in Figure 5.17, the total error for 40 eye patches were low. After this error is higher. The reason is that the test eye patches are more similar to the training patches. In other words, the head pose of a user did not vary much for the eye patches. Table 5.6 also shows the test head pose variation of the user.

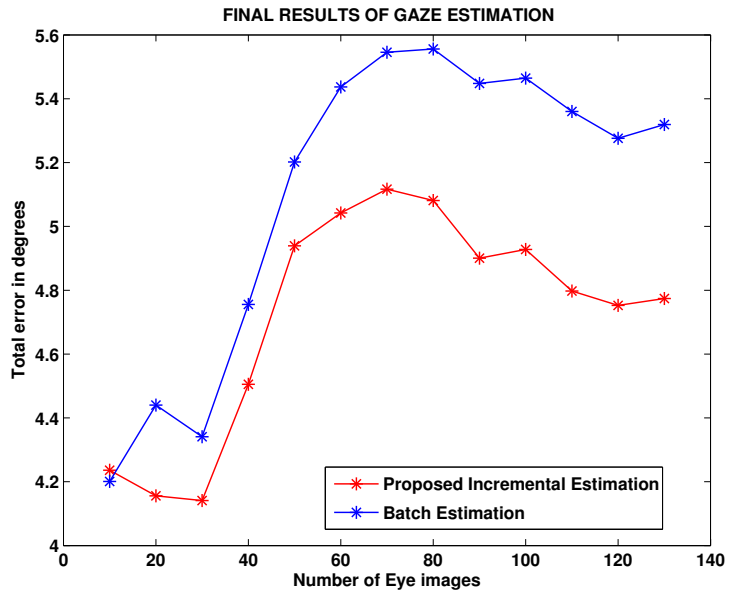



**Figure 5.17:** Performance comparison of baseline model and the model that is learning gaze estimates online under slight head pose.

| Range | Value |
|---|---|
| X-Translation | $-60 \sim 2$ mm |
| Y-Translation | $94 \sim 122$ mm |
| Z-Translation | $808 \sim 837$mm |
| Pitch | $-11° \sim 2°$ |
| Yaw | $-9° \sim 14°$ |
| Roll | $-7° \sim 11°$ |

**Table 5.6:** Head pose Values and Ranges.

The performance of initial calibration stage and online learning of gaze estimates for other subjects is also given in Table 5.7. As can be seen in Table 5.7, the proposed online method is performing better than the batch approach.

The comparison of baseline and online gaze direction parameter estimation under free head pose were evaluated and are given in Figure 5.18. Online parameter estimation using both training and testing samples results in higher accuracy. Figure provides

total error of gaze estimates in degrees for every 20 frames. There is a clear improvement in gaze direction estimation errors. This proves the robustness and more accurate parameter estimates using the proposed method. The variation of head pose values are also reported in Table 5.8. The head pose parameters were estimated using detected landmarks on the face and then the distortion on eye appearance was eliminated using these estimated pose parameters.

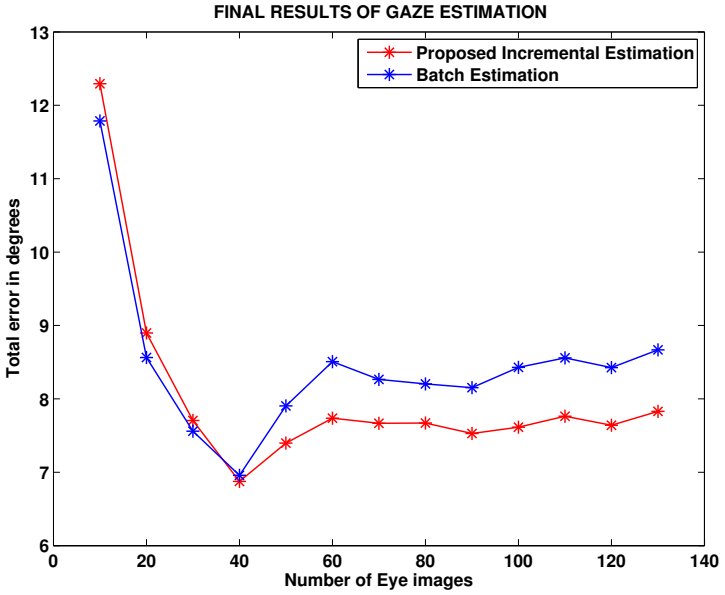



**Figure 5.18:** Performance comparison of baseline model and the model that is learning gaze estimates online under free head pose.

| Range | Value |
|---|---|
| X-translation | $-244 \sim 157$ mm |
| Y-translation | $-11 \sim 143$ mm |
| Z-translation | $489 \sim 867$mm |
| Pitch | $-23° \sim 10°$ |
| Yaw | $-21° \sim 22°$ |
| Roll | $-18° \sim 22°$ |

**Table 5.8:** Head pose Values and Ranges.

The performance of baseline and online models for other subjects is also given in Table 5.9 showing the proposed online method is performing better than the batch approach.

| **METHOD** | **Online Linear SVR** | **Batch Linear SVR** | **Training Samples** |
|---|---|---|---|
| **Subject 1** | 3.91 | 3.95 | 60 training samples. |
| **Subject 2** | 3.0 | 2.98 | |
| **Subject 3** | 4.50 | 4.81 | |
| **Average** | **3.80** | **3.91** | |

**Table 5.7:** Estimation accuracy under slight head motion. Average mean errors and mean angular errors for four subject, measured in degrees.

| METHOD | Online Linear SVR | Batch Linear SVR | Training Samples |
|---|---|---|---|
| **Subject 1** | 8.03 | 8.7 | 60 training samples. |
| **Subject 2** | 6.29 | 6.48 | |
| **Subject 3** | 7.68 | 7.84 | |
| **Average** | **7.33** | **7.67** | |

**Table 5.9:** Estimation accuracy under free head motion. Average mean errors and mean angular errors for four subject, measured in degrees.

### 5.4.7 Adaptive Incremental learning for person independent Gaze estimation

The performance of the proposed method is evaluated for person independent cases. A model is generated for one subject and used for another subject. The aim of the evaluation is to show that the model updates itself according to new user. It is expected that the error of the gaze direction parameter estimates decreases. The performance comparison of baseline and online gaze direction parameter estimation is given in Figure 5.19. It is clear that the model updates itself to the new user and error values decreases. Pose values and ranges of a test subject are reported in Table 5.10. The head pose variation of the samples can be seen in the histograms that are given in Figure 5.20.

Performance of online model update for other subjects is given in Table 5.11. The model is generated for first subject and used for other subjects. The model is updated for other subjects online. The results were reported with/without adapting models. Model update results in high accuracy.

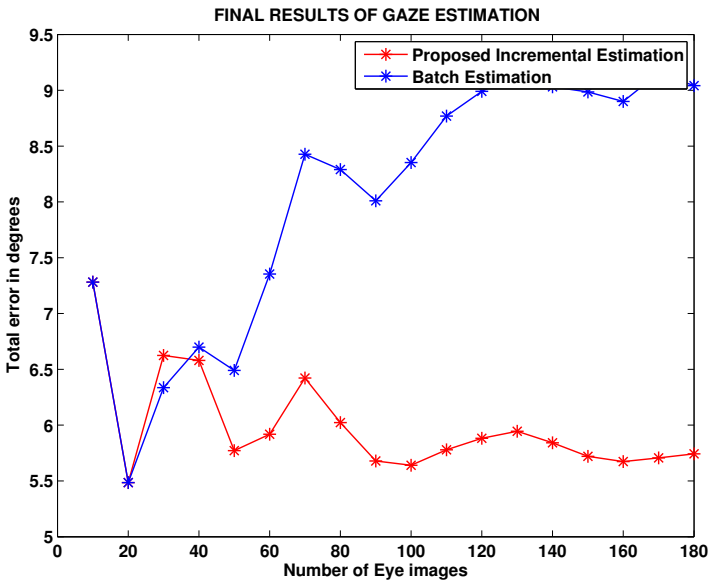



**Figure 5.19:** Performance comparison of baseline model and the model adaptation for person independent gaze estimation.

| Range | Value |
|---|---|
| X-trans. | $-60 \sim 2$ mm |
| Y-trans. | $94 \sim 122$ mm |
| Z-trans. | $808 \sim 837$mm |
| Pitch | $-11° \sim 2°$ |
| Yaw | $-9° \sim 14°$ |
| Roll | $-7° \sim 11°$ |

**Table 5.10:** Head pose Values and Ranges.

The proposed person independent method is different than previously proposed methods. In [121], the authors proposed person independent gaze estimation using an RGB-D camera. In their study, they propose three approaches for modeling the person independent gaze estimation. First, they created several person specific models during training and they selected the best models for a test person during evaluation. They also propose to combined models and generate generic model for all users. However, their approaches require calibration process for each person. On the other hand, our method is based on model adaptation and allows less calibration time. In [46], large synthetic data is generated and then the random forest based regression is used to map eye patches to gaze direction parameters. This method required higher computation power than the proposed method. Second, storing a large synthetic data in a memory might be problem-

| **Proposed Online Learning** | **Subject 1** | **Subject 2** | **Subject 3** | **Average** |
|---|---|---|---|---|
| **Without** | 5.31 | 9.05 | 10.07 | **8.14** |
| **With** | 5.08 | 5.90 | 8.65 | **6.54** |

**Table 5.11:** Performance of online model update for other subjects. The model is generated for one subject and used for other subjects. The model is updated for other subjects online.

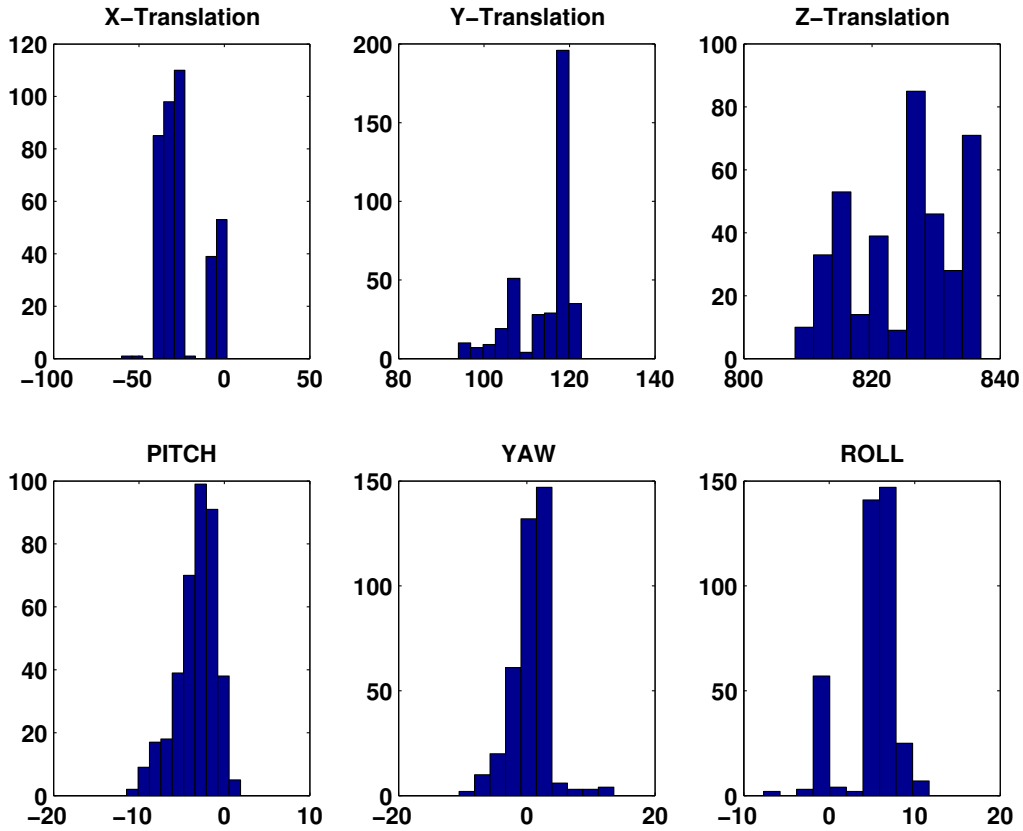



**Figure 5.20:** Head pose variation of test samples for person independent gaze tracking evaluation.

atic for some applications such as mobile applications. In contrast, our method allows similar accuracy using less training samples.

### 5.4.8 Further Evaluation of Online Gaze Estimation

During the initial calibration stage, a set of calibration points was shown to a user. During the online update stage, the calibration points which were not used for training were shown randomly on the screen in order to capture the test eye patches. Locations of calibration points were different than the locations of test points. The evaluation is also performed on more test eye patches.

Figure 5.21 shows both locations of calibration points and the test points. Table 5.12 shows the performance evaluation of baseline and the proposed online method. The accuracy is reported in terms of the angular error in degrees. The results were

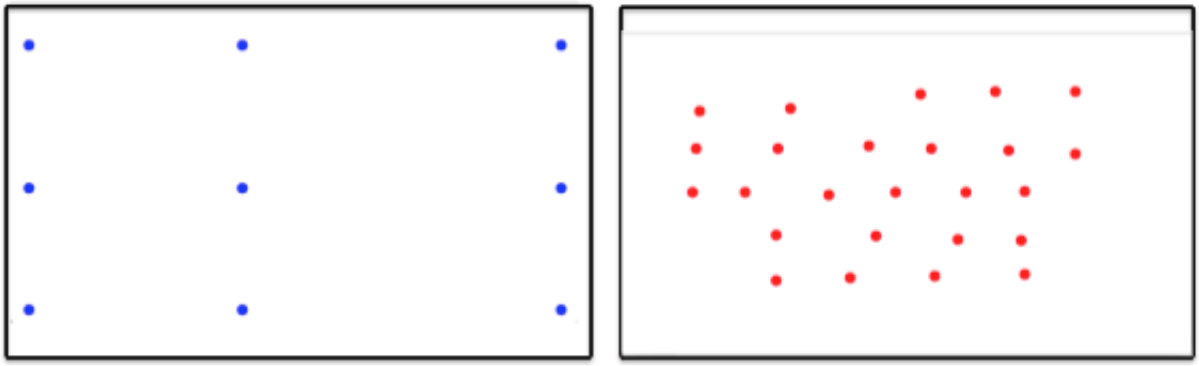


**Figure 5.21:** The locations of the calibration points on the left (blue color) and the locations of test points on the right (red color).

| Proposed Online Learning | Average |
|---|---|
| **Without** | **4.01** |
| **With** | **3.48** |

**Table 5.12:** Performance comparison of baseline model and the model that is learning gaze estimates online under slight head pose.

obtained using 9 calibration points and 60 eye patches during training. During testing, 245 test eye patches were used.

We also report average angular error for every 20 frames (20,40,60,...) in Figure 5.22 together with head pose variation of the user in Table 5.13.

We are providing locations of test and the estimated screen points in Figure 5.23. The accurate estimation of gaze points can be seen on screen images containing the test and the estimated points.

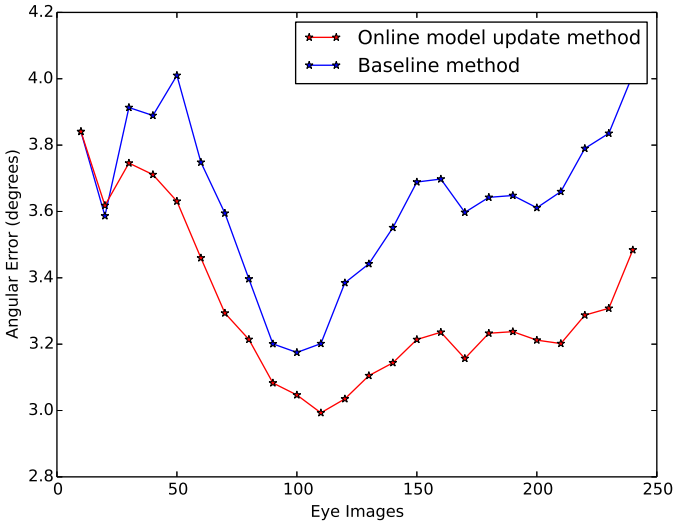



**Figure 5.22:** Performance comparison of baseline model and the model adaptation.

| Range | Value |
|---|---|
| X-trans. | $-55 \sim -29$ mm |
| Y-trans. | $77 \sim 89$ mm |
| Z-trans. | $716 \sim 735$mm |
| Pitch | $-4° \sim 5°$ |
| Yaw | $-12° \sim 2°$ |
| Roll | $-3° \sim 2°$ |

**Table 5.13:** Head pose Values and Ranges.

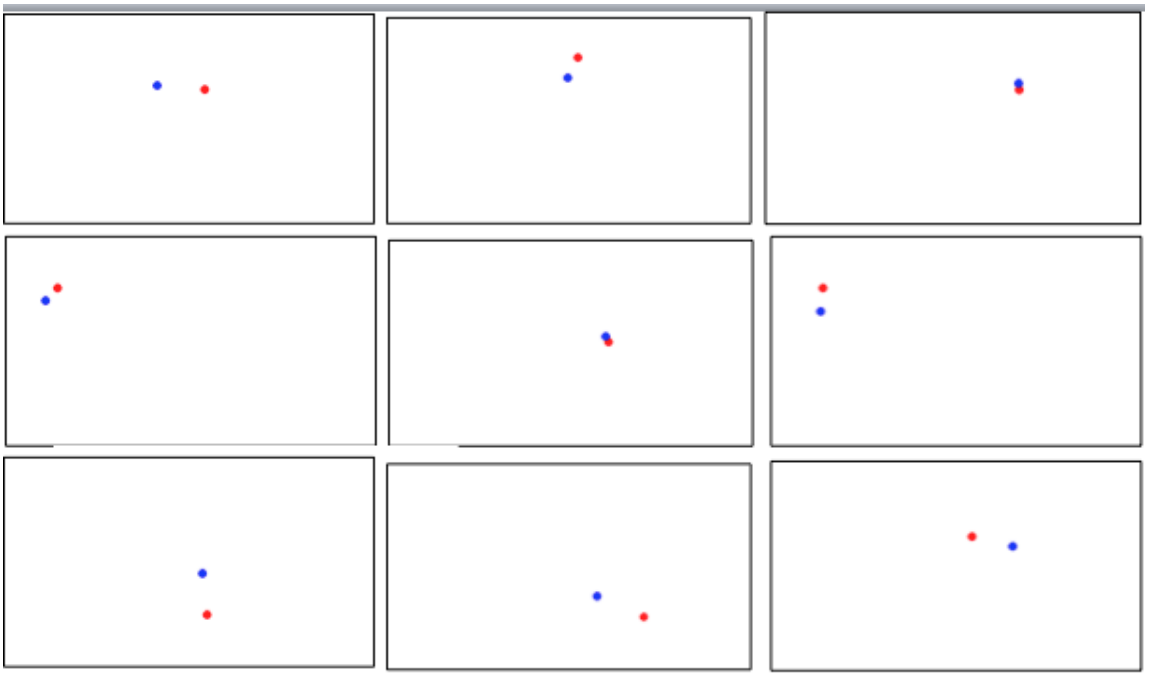


**Figure 5.23:** Estimation of gaze point on the screen. Blue point is the ground truth and the red point is the estimated gaze point.

### 5.4.9 Comparison with the state of the art

The extensive experimental evaluations of appearance based gaze tracking methods under free head pose were conducted on data of several subjects. As it can be seen, we achieved results that are comparable to state of the art methods. There are a number of advantages of the proposed methods. First, the state of the art results, [45, 120] are obtained using commercial gaze tracking machine, [1] for head pose estimation. The system's pose parameter estimation accuracy has not been reported in their evaluations. The landmark based head pose parameter estimation might allow limited range of head orientation and translation of the users. However, such issues are not discussed in their reports. By contrast, the pose parameter estimation of this system is based on estimating landmarks or processing the depth data of the Kinect camera and then estimating the head pose parameters (Section 5.1.2). On the other hand, the head pose estimation methods proposed in this study allow the users to move freely in front of the camera and perform well within a satisfactorily large head pose range. Second, in [45, 120] the translation variation compensation is performed by learning the translational bias using Gaussian regression. However, that regression method requires a substantial amount of samples, 5 minutes video. In contrast, the translation bias is learned by using tensor based regression which is known as a robust regression method for sparse data. This regression method allowed fewer training samples, that is 133 samples. Finally, the proposed methods also based on eye appearance normalization using texture mapping onto the depth data. This allows to eliminate higher head orientation variation.

## 5.5 Discussion

The head pose parameter calculation can be performed in two different ways on the captured intensity images and depth data of the RGB-D camera. First, head pose parameter calculation can be carried out on only depth data. Such a method is described in the previous chapter. The user should be approximately 0.9 meters away from the screen when the head pose parameters are estimated using Kinect depth data. The reason is that the training of the regression forest was performed on the training data captured about 0.9 meter. As the resolution of the eye patches is decreasing, the gaze estimates might not be accurate. Second, the head pose parameters can also be estimated from the detected facial landmarks. In this case, estimated landmarks and the mean face landmarks are used to estimated the head rotation and translation with respect to mean

face. The main advantage of estimating parameters from landmarks is that the user can approach closer to the screen, i.e 0.5 meter. This allows capturing higher resolution eye patches. After experimental observations, head pose parameter estimation is carried out using estimated landmarks and the mean face in our experiments.

The reported results above showed that accurate gaze estimates can be obtained by the head orientation and translation compensation. In particular, gaze direction estimation error reported 6.7° degrees without any compensation. This error is reduced to 5.1° when the head orientation compensation is applied. Angular error is further reduced to $\sim 3°$ when head orientation and translation compensation is applied. Although $\sim 3°$ has been achieved by employing compensation techniques, this accuracy might not be sufficient for some applications. Further error reduction can be achieved by addressing alignment problems which occur during head orientation and translation.

## 5.6 Conclusion

Estimates of gaze direction parameters are obtained under slight head pose and then the proposed method used to learn and compensate gaze estimates online under head pose. Furthermore, the proposed approach provides gaze estimates from images acquired from a low cost device. The experimental results shows that the proposed online adaptive learning using images acquired from a low cost device can achieve about three degrees angle accuracy.

The proposed appearance based eye gaze estimation method and the consumer eye gaze tracker were compared by conducting extensive experiments. The experimental results showed that the appearance based system and the consumer based eye gaze tracker resulted in similar performance results under fix head pose. Furthermore, the performance of the eye gaze tracker was not affected natural head pose. However, by the proposed appearance based method was also robust under natural head pose.

# Chapter 6

# Conclusions and Future Work

Gaze tracking technology has been studied for many years. Although progress has been reported in previously published studies, robust and accurate identity invariant gaze tracking technology under free head pose motion is still an unsolved problem. Another disadvantage of the available gaze technologies is that the hardware is unaffordable for many people and this causes limited usage of this technology. Therefore, the main focus of the research is the development of gaze tracking methods using low cost available sensors such as a webcam and Kinect.

This thesis has focused on developing methods for the real-time appearance-based gaze estimation under free head pose using a low cost Kinect camera. The proposed methods addressed the existing problems of the appearance-based gaze estimation. As it has been reported in the previous chapters, the main problem was that the accuracy of the appearance-based methods degrades when users move from an initial calibration position to a new position. This problem has been addressed by combining the proposed methods for the head pose estimation and the proposed the appearance-based gaze estimation methods. First, a method was proposed to eliminate distortion due to head pose variation on eye appearances and then these eye images were used as inputs to regression models. Eliminating distortion due to head orientation has been achieved using perspective projection. The perspective projection was calculated using the estimated head pose parameters. Translation distortion was also eliminated by learning the bias between actual values of gaze points and the estimated gaze points. The experimental results have showed that eliminating distortion due to head pose variation and translation resulted in the better estimates of the gaze points on the screen. Second, a novel method which allows gaze direction parameter learning online has been proposed. The proposed method allowed for updating models online to deal with new eye image

appearances. The novel proposed method is based on generating the Gaussian Mixture model (GMM) of the training data and online Support Vector Regression (OSVR). The training data is also modelled using eigenspace decomposition for computational purposes. a GMM model allows detecting any changes on eye image appearances and updating OSVR models for these new appearances online. In this way, the recalibration process has been eliminated when a user moves from an initial calibration position to a new position.

The techniques were proposed to achieve the head pose estimation using Kinect. The proposed method allowed employing RGB images and depth data for estimating head pose parameters accurately. The proposed method was multimodal random forest based tensor regression. This novel method has been developed by extending random forests in three ways: (i) by using tensor-based regression at each leaf node, (ii) by fusing depth and intensity data using tensor regression at each leaf node and (iii) fusing RGB and depth data using random forests. As a result, a new method based on multi-modal random forests and tensor models is proposed for more accurate and robust head pose estimation. The random forests allow modeling large head pose while tensor models provide accurate results.

## 6.1 Conclusions

Conclusions can be drawn by considering experimental evaluations of the proposed unified system. This unified system is based on the proposed appearance-based gaze and head pose estimation methods.

The method which allows mapping eye images to gaze points provides good estimates when a user positions himself or herself in the calibration position. If the user moves away from the camera, the proposed method also allows for accurate estimates. The performance of this method does not degrade very much since it allows elimination of head orientation and translation distortion. However, the biases between estimated gaze and the actual gaze points need to be learned to eliminate translation bias. This requires additional calibration time. A long calibration time is not preferable for the gaze tracking systems.

The second proposed method which is based on online gaze parameter learning allows better gaze estimates in real-time. The proposed method is based on modeling the initial

calibration data by a Gaussian Mixture Model (GMM) and then updating online Support Vector Regression (OSVR). This system allows users to change their positions without recalibration. This method requires less calibration time than the first proposed method.

The proposed head pose method which is used in conjunction with the proposed gaze estimation methods has several advantages. First, the method provides head pose parameters of a user under large head pose variation and occlusion. Second, the method allows for fast head pose parameter computation (real-time) without the need of a GPU. Furthermore, the method is user independent. The performance comparisons of the proposed method and the other methods showed that the proposed method is robust and accurate and that the proposed method was demonstrated on the available Biwi Kinect Head Pose Database. The experiments showed that the proposed framework that combines random forests and tensor regression outperforms typical random forests.

## 6.2 Future Work

Further extensions of the proposed appearance based gaze estimation methods in this chapter while taking the research study presented in this thesis as a basis are outlined in this section.

The methods described in chapter 4 can be extended in several ways. First, the number of trees in the forest can be increased. More trees might model the data more precisely for more accurate head pose parameter estimation. Second, the performance of feature fusion might be investigated more by exploiting different features inside the random regression forest framework. Furthermore, the performance of different feature fusion can also be investigated at the leaf nodes of the random forest. Finally, new regression techniques might be introduced instead of tensor based regression.

The proposed methods in chapter 5 can also be extended in several ways. The appearance based eye gaze estimator without personal calibration under free head pose can be studied. The gaze parameter learning can be achieved while the user is watching a video. Since the user focuses on specific points, such as faces on the video, these locations can be estimated using saliency. Saliency of an image is usually calculated using image features on the image. Recent studies also showed that people's attention is around the faces. The estimator can be generated by capturing eye images of a person while the person is looking at a number of images or watching a video. Enhancement

of the saliency map can also be investigated by aggregating saliency maps according to similarity of the captured eye images. These aggregated saliency maps can be used as the probability distributions of the gaze points. Gaze probability maps provide approximate estimates of gaze points which can be used to map onto eye images. A regression function can be learned from eye images and estimated gaze points.

# Bibliography


[1] "http://www.tobii.com"

[2] A. Fathi, Y. Li, and J. M. Rehg, "Learning to recognize daily actions using gaze," in *Proc. European Conf. Computer Vision*, pp. 314–327, 2012.

[3] E. D. Guestrin and M. Eizenman, "Remote point-of-gaze estimation requiring a single-point calibration for applications with infants," in *Symp. Eye Tracking Research and Applications*, pp. 267–274, 2008.

[4] J. Chen and Q. Ji, "Probabilistic gaze estimation without active personal calibration," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 609–616, 2011.

[5] N. M. Bakker, B. A. J. Lenseigne, S. Schutte, E. B. M. Geukers, P. P. Jonker, F. C. T. van der Helm, and H. J. Simonsz, "Accurate gaze direction measurements with free head movement for strabismus angle estimation," *IEEE Trans. Biomedical Engineering*, vol. 60, no. 11, pp. 3028–3035, 2013.

[6] J. W. Lee, H. Heo, and K. R. Park, "A novel gaze tracking method based on the generation of virtual calibration points," *Sensors*, vol. 13, no. 8, pp. 10802–10822, 2013.

[7] M. Cerf, J. Harel, W. Einhäuser, and C. Koch, "Predicting human gaze using low-level saliency combined with face detection," in *NIPS*, vol. 20, pp. 241–248, 2008.

[8] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," in *Proc. IEEE Conf. Computer Vision*, pp. 2106–2113, 2009.

[9] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607 –626, 2009.

[10] M. Dantone, J. Gall, G. Fanelli, and V. L.Gool, "Real-time facial feature detection using conditional regression forests," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2578–2585, 2012.

[11] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and V. L. Gool, "Random forests for real time 3d face analysis," *Int. J. Computer Vision*, vol. 101, no. 3, pp. 437–458, 2012.

[12] G. Fanelli, J. Gall, and L. V. Gool, "Real time head pose estimation with random regression forests," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 617–624, 2011.

[13] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2188–2202, 2011.

[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1297–1304, 2011.

[15] A. Criminisi, D. Robertson, E. Konukoglu, J. Shotton, S. Pathak, S. White, and K. Siddiqui, "Regression forests for efficient anatomy detection and localization in computed tomography scans," *Medical Image Analysis*, vol. 17, no. 8, pp. 1293–1303, 2013.

[16] K. C. Lee and D. Kriegman, "Online learning of probabilistic appearance manifolds for video-based recognition and tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 852–859, 2005.

[17] F. Parrella, "Online support vector regression," *Master's Thesis, Department of Information Science, University of Genoa, Italy*, 2007.

[18] G. Fanelli, T. Weise, J. Gall, and L. V. Gool, "Real time head pose estimation from consumer depth cameras," in *Symp. German Association Pattern Recognition*, pp. 101–110, 2011.

[19] M. P. Kassner and W. R. Patera, "Pupil" *MIT Master's Thesis*, 2012.

[20] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 3,

pp. 478–500, 2010.

[21] T. Bar, J. F. Reuter, and J. M. Zollner, "Driver head pose and gaze estimation based on multi-template icp 3-d point cloud alignment," in *Proc. IEEE Conf. Intelligent Transportation Systems*, pp. 1797–1802, 2012.

[22] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation and augmented reality tracking: An integrated system and evaluation for monitoring driver awareness," *IEEE Trans. Intell. Transport. Sys.*, vol. 11, no. 2, pp. 300–311, 2010.

[23] E. Murphy-Chutorian, A. Doshi, and M. M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *Proc. IEEE Conf. Intell. Transport. Sys.*, pp. 709–714, 2007.

[24] E. Murphy-Chutorian and M. Trivedi, "3d tracking and dynamic analysis of human head movements and attentional targets," in *Proc ACM/IEEE Conf. Distributed Smart Cameras*, pp. 1–8, 2008.

[25] E. M. Chutorian and M. M. Trivedi, "Hyhope: Hybrid head orientation and position estimation for vision-based driver head tracking," in *Intelligent Vehicles Symposium*, 2008.

[26] K. H. Tan, D. Kriegman, and N. Ahuja, "Appearance-based eye gaze estimation," in *Proc. IEEE Workshop Applications of Computer Vision*, pp. 191–195, 2002.

[27] C. H. Morimoto and M. R. M. Marcio, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4–24, 2005.

[28] J. Chen, Y. Tong, W. Gray, and Q. Ji, "A robust 3d eye gaze tracking system using noise reduction," in *Symp. Eye Tracking Research and Applications*, pp. 189–196, ACM, 2008.

[29] C. Morimoto, A. Amir, and M. Flickner, "Detecting eye position and gaze from a single camera and 2 light sources," in *Proc. Conf. Pattern Recognition*, vol. 4, pp. 314–317 vol.4, 2002.

[30] K. A. F. Mora and J. M. Odobez, "Geometric generative gaze estimation (g3e) for remote rgb-d cameras," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1773–1780, 2014.

[31] D. Model and M. Eizenman, "An automatic personal calibration procedure for

advanced gaze estimation systems," *IEEE Trans. Biomedical Engineering*, vol. 57, no. 5, pp. 1031–1039, 2010.

[32] E. D. Guestrin and M. Eizenman, "General theory of remote gaze estimation using the pupil center and corneal reflections," *IEEE Trans. Biomedical Engineering*, vol. 53, no. 6, pp. 1124–1133, 2006.

[33] S. W. Shih and J. Liu, "A novel approach to 3-d gaze tracking using stereo cameras," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 34, no. 1, pp. 234 –245, 2004.

[34] D. Beymer and M. Flickner, "Eye gaze tracking using an active stereo head," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 451–8, 2003.

[35] W. G. J. Chen, Y. Tong and Q. Ji, "A robust 3d eye gaze tracking system using noise reduction," in *Symp. Eye Tracking Research and Applications*, pp. 189–196, 2008.

[36] D. Memmert, "The effects of eye movements, age, and expertise on inattentional blindness," *Consciousness and cognition*, vol. 15, no. 3, pp. 620–627, 2006.

[37] J. S. Agustin, H. Skovsgaard, E. Mollenbach, M. Barret, D. W. H. M. Tall, and J. P. Hansen, "Evaluation of a low-cost open-source gaze tracker," in *Symp. Eye-Tracking Research and Applications*, pp. 77–80, 2010.

[38] C. W. Cho, J. W. Lee, K. Y. Shin, E. C. Lee, K. R. Park, H. Lee, and J. Cha, "Gaze detection by wearable eye-tracking and nir led-based head-tracking device based on svr," *ETRI Journal*, vol. 34, no. 4, pp. 542–552, 2012.

[39] J. W. Lee, C. W. Cho, K. Y. Shin, E. C. Lee, and K. R. Park, "3d gaze tracking method using purkinje images on eye optical model and pupil," *Optics and Lasers in Engineering*, vol. 50, no. 5, pp. 736–751, 2012.

[40] S. Baluja and D. Pomerleau, "Non-intrusive gaze tracking using artificial neural networks," tech. rep., 1994.

[41] L. Q. Xu, D. Machin, and P. Sheppard, "A novel approach to real-time non-intrusive gaze finding," in *Proc. Conf. British Machine Vision*, pp. 428–437, 1998.

[42] O. Williams, A. Blake, and R. Cipolla, "Sparse and semi-supervised visual mapping with the s 3gp," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 230 – 237, 2006.

[43] Y. Sugano, Y. Matsushita, Y. Sato, and H. Koike, "An incremental learning method for unconstrained gaze estimation," *Proc. European Conf. Computer Vision*, vol. 13, pp. 656–667, 2008.

[44] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[45] F. Lu, T. Okabe, Y. Sugano, and Y. Sato, "A head pose-free approach for appearance-based gaze estimation," in *Proc. Conf. British Machine Vision*, pp. 126.1–126.11, 2011.

[46] Y. Sugano, Y. Matsushita, and Y. Sato, "Learning-by-synthesis for appearance-based 3d gaze estimation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1821–1828, 2014.

[47] C. Koch and S. Ullman, "Shifts in selective visual attention: Towards the underlying neural circuitry," *Human Neurobiol*, vol. 188, no. 4, pp. 219–227, 1985.

[48] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," *Advances in Neural Information Processing Systems*, pp. 545–552, 2006.

[49] A. Borji and L. Itti, "Exploiting local and global patch rarities for saliency detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 478–485, 2012.

[50] A. Borji, "Boosting bottom-up and top-down visual features for saliency estimation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 438–445, 2012.

[51] W. Kienzle, F. A. Wichmann, B. Scholkopf, and M. O. Franz, "A nonparametric approach to bottom-up visual saliency," pp. 689–696, 2007.

[52] N. D. Bruce and J. K. Tsotsos, "Saliency, attention, and visual search: An information theoretic approach," *Journal of vision*, vol. 9, no. 3, p. 5, 2009.

[53] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[54] Y. Sugano, Y. Matsushita, and Y. Sato, "Appearance-based gaze estimation using visual saliency," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, 2013.

[55] S. Langton, H. H, and E. Tessler, "The influence of head contour and nose angle on the perception of eye-gaze direction," *Perception and psychophysics*, vol. 66, no. 5, pp. 752–771, 2004.

[56] M. Osadchy, Y. L. Cun, and M. L. Miller, "Synergistic face detection and pose estimation with energy-based models," *The Journal of Machine Learning Research*, vol. 8, pp. 1197–1215, 2007.

[57] T. F. Cootes, G. J. Edward, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[58] K. Ramnath, S. Koterba, J. Xiao, C. Hu, I. Matthews, S. Baker, J. Cohn, and T. Kanade, "Multi-view aam fitting and construction," *Int. J. Computer Vision*, vol. 76, no. 2, pp. 183–204, 2008.

[59] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proc. Conf. Computer Graphics and Interactive Techniques*, pp. 187–194, 1999.

[60] M. Storer, M. Urschler, and H. Bischof, "3d-mam: 3d morphable appearance model for efficient fine head pose estimation from still images," in *Proc. IEEE Conf. Computer Vision Workshops*, pp. 192–199, 2009.

[61] D. Cristinacce and T. Cootes, "Feature detection and tracking with constrained local models," in *Proc. Conf. British Machine Vision*, vol. 3, pp. 929–938, 2006.

[62] W. Guo, I. Kotsia, and I. Patras, "Tensor learning for regression," *IEEE Trans. Image Processing*, vol. 21, no. 2, pp. 816–827, 2012.

[63] J. M. Odobez and S. O. Ba, "Evaluation of multiple cues head pose tracking algorithm in indoor environments," in *Proc. Conf. Multimedia and Expo*, 2005.

[64] M. L. Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 322–336, 2000.

[65] N. Gourier, D. Hall, and J. L. Crowley, "Estimating face orientation from robust detection of salient facial structures," in *FG Net Workshop on Visual Observation of Deictic Gestures*, pp. 1–9, 2004.

[66] T. Vatahska, M. Bennewitz, and S. Behnke, "Feature-based head pose estimation

from images," in *Proc. Conf. Humanoid Robots*, pp. 330–335, 2007.

[67] J. Whitehill and J. Movellan, "A discriminative approach to frame-by-frame head pose tracking," in *Proc. IEEE Conf. Automatic Face and Gesture Recognition*, pp. 1 –7, 2008.

[68] M. D. Breitenstein, D. Kuettel, T. Weise, L. V. Gool, and H. Pfister, "Real-time face pose estimation from single range images," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[69] E. Seemann, K. Nickel, and R. Stiefelhagen, "Head pose estimation using stereo vision for human-robot interaction," in *Proc. IEEE Conf. Automatic Face and Gesture Recognition*, pp. 626 – 631, 2004.

[70] L. P. Morency, P. Sundberg, and T. Darrell, "Pose estimation using 3d view-based eigenspaces," in *Proc. IEEE Conf. Analysis and Modeling of Faces and Gestures Workshops*, pp. 45 – 52, 2003.

[71] T. Weise, H. Li, L. V. Gool, and M. Pauly, "Face/off: Live facial puppetry," in *Symp. SIGGRAPH/Eurographics Computer animation*, pp. 7–16, ACM, 2009.

[72] T. Weise, S. Bouaziz, H. Li, and H. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graphics*, vol. 30, no. 4, p. 77, 2011.

[73] S. Mehryar, K. Martin, K. N. Plataniotis, and S. Stergiopoulos, "Automatic landmark detection for 3d face image processing," in *IEEE Congress Evolutionary Computation*, pp. 1–7, 2010.

[74] Y. Wang, C. S. Chua, and Y. Ho, "Facial feature detection and face recognition from 2d and 3d images," *Pattern Recognition Letters*, vol. 23, no. 10, pp. 1191–1202, 2002.

[75] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[76] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning," *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*, vol. 5, no. 6, p. 12, 2011.

[77] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Comput.*, vol. 9, no. 7, pp. 1545–1588, 1997.

[78] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1022–1029, 2009.

[79] J. Gall, N. Razavi, and L. V. Gool, "On-line adaption of class-specific codebooks for instance tracking," in *Proc. Conf. British Machine Vision*, pp. 1–2, 2010.

[80] A. Yao, J. Gall, and L. Van Gool, "A hough transform-based voting framework for action recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2061–2068, 2010.

[81] R. Maree, P. Geurts, J. Piater, and L. Wehenkel, "Random subwindows for robust image classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 34–40, 2005.

[82] F. Moosmann, B. Triggs, F. Jurie, *et al.*, "Fast discriminative visual codebooks using randomized clustering forests," *Advances in Neural Information Processing Systems 19*, pp. 985–992, 2007.

[83] F. Schroff, A. Criminisi, and A. Zisserman, "Object class segmentation using random forests," in *Proc. Conf. British Machine Vision*, pp. 54.1–54.10, 2008.

[84] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[85] J. Winn and J. Shotton, "The layout consistent random field for recognizing and segmenting partially occluded objects," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 37–44, 2006.

[86] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 775–781 vol. 2, 2005.

[87] H. T. Chen, T. L. Liu, and C. S. Fuh, "Segmenting highly articulated video objects with weak-prior random forests," in *Proc. European Conf. Computer Vision*, pp. 373–385, 2006.

[88] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof, "Prost: Parallel robust online simple tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 723–730, 2010.

[89] Z. Lin, Z. Jiang, and L. S. Davis, "Recognizing actions by shape-motion prototype

trees," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 444–451, 2009.

[90] K. Mikolajczyk and H. Uemura, "Action recognition with motion-appearance vocabulary forest," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[91] K. K. Reddy, J. Liu, and M. Shah, "Incremental action recognition using feature-tree," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1010–1017, 2009.

[92] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[93] C. C. Chang and C. J. Lin, "Training v-support vector classifiers: theory and algorithms," *Neural computation*, vol. 13, no. 9, pp. 2119–2147, 2001.

[94] L. Wolf, H. Jhuang, and T. Hazan, "Modeling appearances with low-rank svm," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1–6, 2007.

[95] H. Pirsiavash, D. Ramanan, and C. Fowlkes, "Bilinear classifiers for visual recognition," in *Advances in Neural Information Processing Systems*, pp. 1482–1490, 2009.

[96] D. Tao, X. W. X. Li, and, W. Hu, and S. J. Maybank, "Supervised tensor learning," *Knowledge and Information Systems*, vol. 13, no. 1, pp. 1–42, 2007.

[97] R. Tomioka and K. Aihara, "Classifying matrices with a spectral regularization," in *Proc. ACM Conf. Machine learning*, pp. 895–902, 2007.

[98] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[99] S. O. Ba and J. M. Odobez, "Evaluation of multiple cues head pose tracking algorithm in indoor environments," in *Proc. Conf. Multimedia and Expo*, 2005.

[100] T. P. G. Cauwenberghs, and, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, pp. 409–415, 2001.

[101] M. Mario, "On-line support vector machine regression," in *Proc. European Conf. Machine Learning*, pp. 282–294, 2002.

[102] J. Ma, J. Theiler, and S. Perkins, "Accurate on-line support vector regression," *Neural Computation*, vol. 15, no. 11, pp. 2683–2703, 2003.

[103] X. H. S. Mao and M. Wang, "Image jacobian matrix estimation based on online support vector regression.," *Int J Adv Robotic*, vol. 9, no. 111, 2012.

[104] M. Kristan and A.Leonardis, "Online discriminative kernel density estimation," in *Proc Conf. Pattern Recognition*, pp. 581–584, 2010.

[105] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 49–56, 2010.

[106] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems 13*, pp. 409–415, 2001.

[107] L. Ralaivola and F. d'Alch Buc, "Incremental support vector machine learning: a local approach," in *In Proceedings of ICANN*, pp. 322–329, 2001.

[108] P. M. Hall, D. Marshall, and R. R. Martin, "Incremental eigenanalysis for classification," in *Proc. Conf. British Machine Vision*, pp. 286–295, 1998.

[109] D. Skocaj and A. Leonardis, "Weighted and robust incremental method for subspace learning," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1494–1501, 2003.

[110] M. A.Turk and A. P.Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 586–591, 1991.

[111] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," tech. rep., 1998.

[112] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[113] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *Proc. IEEE Conf. International Advanced Video and Signal Based Surveillance*, pp. 296–301, 2009.

[114] L. P. Morency, "3d constrained local model for rigid and non-rigid facial tracking,"

in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2610–2617, 2012.

[115] J. M. Saragih, S. Lucey, and J. F. Cohn, "Deformable model fitting by regularized landmark mean-shift," *Int. J. Comput. Vision*, vol. 91, no. 2, pp. 200–215, 2011.

[116] L. P. Morency, J. Whitehill, and J. Movellan, "Generalized adaptive view-based appearance model: Integrated framework for monocular head pose estimation," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pp. 1–8, 2008.

[117] X. Xiong and F. D. L. Torre, "Supervised descent method and its applications to face alignment," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 532–539, 2013.

[118] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," in *Proc. Conf. Computer Graphics and Interactive Techniques*, pp. 39–46, 1995.

[119] R. Szeliski, "Image mosaicing for tele-reality applications," in *Proc. IEEE Workshop Applications of Computer Vision*, pp. 44–53, 1994.

[120] F. Lu, T. Okabe, Y. Sugano, and Y. Sato, "Learning gaze biases with head motion for head pose-free gaze estimation," *Image and Vision Computing*, vol. 32, no. 3, pp. 169–179, 2014.

[121] M. Funes, A. Kenneth, and J. Odobez, "Person independent 3d gaze estimation from remote rgb-d cameras," in *Proc. Conf. Image Processing*, 2013.